

# THE NATURE OF RETROGRADE ANALYSIS FOR CHINESE CHESS <sup>1</sup>

Haw-ren Fang<sup>2</sup>

Maryland, USA

## ABSTRACT

Retrograde analysis has been successfully applied to solve Awari (Romein and Bal, 2003), and construct 6-piece Western chess endgame databases (Thompson, 1996). However, its application to Chinese chess is limited because of the *special rules* about indefinite move sequences.

In (Fang, Hsu, and Hsu, 2004), problems caused by the most influential rule, *checking indefinitely*<sup>3</sup>, were successfully solved in practical cases, with 50 selected endgame databases constructed in accord with this rule, where the 60-move-rule was ignored. Other special rules have much less impact on contaminating the databases, as verified by the rule-tolerant algorithms (Fang, 2004). For constructing complete endgame databases, we need rigorous algorithms. There are two rule sets in Chinese chess: Asian rule set and Chinese rule set. In this paper, an algorithm is successfully developed to construct endgame databases in accord with the Asian rule set. The graph-theoretical properties are also explored as well.

## 1. INTRODUCTION

Retrograde analysis is widely applied to construct databases of finite, two-player, zero-sum and perfect information games (van den Herik, Uiterwijk, and van Rijswijck, 2002). The classical algorithm first determines all terminal positions (e.g., checkmate or stalemate in both Western chess and Chinese chess), and then iteratively propagates the values back to their predecessors until no propagation is possible. The remaining undetermined positions are then declared as draws in the final phase.

In Western chess, as in many other games, if a game continues endlessly without reaching a terminal position, the game ends in a draw. However, in Chinese chess, there are special rules other than checkmate and stalemate to end a game. The endgame databases of Chinese chess constructed by retrograde analysis may have errors if the special rules are not taken into account. Nevertheless, the endgame databases, in which only one side has attacking pieces, are not affected by these special rules (Fang, Hsu, and Hsu, 2000). Using this fact, 151 endgame databases with attacking pieces on one side only are correctly constructed (Fang *et al.*, 2000; Wu and Beal, 2001).

The most influential special rule is non-mutual checking indefinitely. If only one player checks his<sup>4</sup> opponent continuously without ending, he loses the game. Problems caused by the rule of checking indefinitely are practically solved (Fang, Hsu, and Hsu, 2002; Fang *et al.*, 2004), with the 50 endgame databases successfully constructed in accord with this rule (Fang *et al.*, 2004). These databases are selected potentially

---

<sup>1</sup>I began working on Chinese chess endgame databases by retrograde algorithms in mid 1996 under the direction of my master's thesis advisor, Shun-Chin Hsu, in National Taiwan University. I noticed the problems caused by the special rules, particularly the rule of checking indefinitely. A couple of different approaches had been used without success. After graduation, I did not stop thinking about this problem while I fulfilled military service. One summer evening in 1998, I was tackling this problem again in the office when everybody else went out for dinner. In a flash, I found the algorithm to compute the maximum move pattern of checking indefinitely, which was a prelude of all the algorithms and theories developed afterward. The weather in Taiwan is often cloudy, but when I walked out of my office on that evening, I saw a blood red sky as a prophecy for a big change in the weather.

<sup>2</sup>Department of Computer Science, University of Maryland, A.V. Williams Building, College Park, Maryland 20742, USA. Email: hrfang@cs.umd.edu.

<sup>3</sup>Another name of the concept of checking indefinitely is *perpetual checking*.

<sup>4</sup>In this article we use 'his' and 'he' when both 'her/his' and 'she/he' are possible.

contaminated by the special rules. We call a database *complete* if it is not contaminated by any special rules. There are two rule sets used in Chinese chess: Asian rule set and Chinese rule set. Out of these 50 databases, 24 are verified complete with the Asian rule set, whereas 21 are verified complete with the Chinese rule set (Fang, 2004). The next step is to correct the incomplete databases.

The Asian rule set is used generally in the world major tournaments, local competitions other than those in China, and computer Chinese chess competitions, whereas the Chinese rule set is used only in China. Therefore, the Asian rule set deserves more attention than the Chinese rule set. In this paper, we focus on how to build the endgame databases in accord with the Asian rule set.

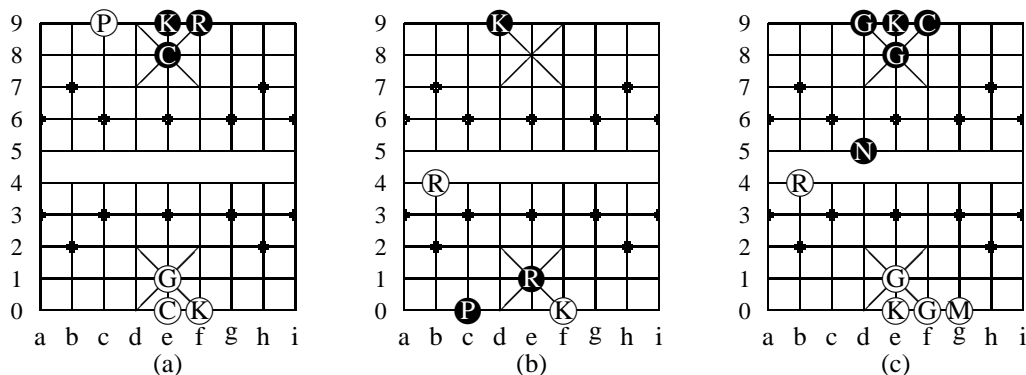
The organization of this article is as follows. Section 2 describes and abstracts the Chinese-chess special rules in the Asian rule set. Section 3 discusses the essential move patterns of checking/chasing indefinitely. Section 4 gives an algorithm to build complete win-draw-loss endgame databases of Chinese chess. Section 5 gives an algorithm to construct infallible endgame databases in accord with the Asian rule set. Concluding remarks are given in Section 6.

## 2. SPECIAL RULES IN CHINESE CHESS

Retrograde algorithms are widely used to construct the databases of finite, two-player, zero-sum and perfect-information games (van den Herik *et al.*, 2002). In Western chess, a non-drawn game ends in checkmate or by a resignation; a drawn game can be the result of a stalemate, repetition of positions, or insufficient material to checkmate. In Chinese chess, there are other special rules to end a game.

### 2.1 A Brief Overview of the Special Rules

In Chinese chess, the two sides are called Red and Black. Each side has one King, two Guards, two Ministers, two Rooks, two Knights, two Cannons, and five Pawns, which are abbreviated as K, G, M, R, N, C and P, respectively<sup>5</sup>. The pieces Rook, Knight, Cannon, and Pawn are called *attacking pieces* since they can move across the *river*, the imaginary stream between the two central horizontal lines of the board. In contrast, Guards and Ministers are called *defending pieces* because they are confined in the domestic region<sup>6</sup>. A *position* in Chinese chess is an assignment of a subset of pieces to distinct addresses on the board with a certain player-to-move.



**Figure 1:** Examples of (a) mutual checking indefinitely, (b) non-mutual chasing indefinitely, and (c) mutual chasing indefinitely.

A game ends in any position of checkmate or stalemate. In addition, there are other end positions determined by the special rules of indefinite move sequences. All the special rules discussed in this paper refer to the rule book (Association, 1999). An indefinite move sequence is conceptually an infinite move sequence. In real games, it is determined by the threefold repetition of positions in a finite move sequence (Association,

<sup>5</sup>The English translation of the Chinese names differs by author.

<sup>6</sup>Information on Chinese chess such as notation and basic rules in English can be found in the ICGA web page of Chinese chess <http://www.cs.unimaas.nl/icga/games/chinesechess/>, and in FAQ of the Internet news group [rec.games.chinese-chess](http://www.chessvariants.com/chinfaq.html), which is available at <http://www.chessvariants.com/chinfaq.html>.

1999, page 65, rule 3). An indefinite move sequence is composed of two *semi-sequences*: one consists of the moves by Red and the other has the moves by Black.

In Chinese chess, a special rule is that if only one player checks his opponent continuously (i.e., without ending), then he loses. This rule is called *non-mutual checking indefinitely*. In real games, this means that a player loses if he cannot prevent his King from being captured without checking his opponent indefinitely. If both players check each other continuously without ending, the game ends in a draw. This rule is called *mutual checking indefinitely*. An example is shown in Figure 1(a) where Red is to move. Both players are forced to check each other cyclically with the moves Ge1-f2, Ce8-f8, Gf2-e1, Cf8-e8, etc.

There are special rules about *chasing indefinitely* in Chinese chess. The general concept is that a player cannot chase some opponent's piece continuously without ending (Association, 1999, page 64). The term *chase* is defined similarly to the term check, but the prospective piece to be captured is not the King but another piece. A move sequence consists of two semi-sequences: one played by Red and the other by Black. A semi-sequence of chasing moves is either *allowed* or *forbidden*. A forbidden semi-sequence of chasing moves in the Chinese rule set may be allowed in the Asian rule set. As pointed out in (Fang, 2004), the complicated rules of chasing indefinitely cause the difficulty to adapt the algorithms for checking indefinitely in (Fang *et al.*, 2004) for chasing indefinitely.

Recall that we address in the Asian rule set in this paper. The Asian rule set is summarized as follows (Association, 1999, page 64, section 2).

1. If both players check each other indefinitely, the game ends in a draw.
2. If only one player checks the other indefinitely, the player who checks loses the game.
3. Otherwise, if only one player plays a forbidden semi-sequence of chasing moves, he loses the game.
4. Otherwise, the game ends in a draw.

With the above summary, both mutual checking indefinitely and mutual chasing indefinitely result in a draw. If one player checks indefinitely and the other chases indefinitely, the one who checks loses the game. In Figure 1(b) is an example of non-mutual chasing indefinitely, where Red is forced to chase the Black Pawn endlessly with the moves Rb4-c4, Pc0-b0, Rc4-b4, Pb0-c0, etc. In Figure 1(c), if the game continues indefinitely with the moves Rb4-b5, Nd5-c3, Rb5-c5, Nc3-e4, Rc5-c4, Ne4-d6, Rc4-d4, Nd6-f5, etc., the game ends in a draw because of mutual chasing indefinitely (Association, 1999, page 89, rule 22).

## 2.2 Abstracting Special Rules

A two-player, finite and zero-sum with perfect information game such as Western chess and Chinese chess can be represented as a *game graph*  $G = (V, E)$ , which is directed, bipartite and possibly cyclic, where  $V$  is the set of vertices and  $E$  is the set of edges. Each *vertex* indicates a position. Each *directed edge* corresponds to a move from one position to another, with the relationship of *parent* and *child* respectively. Positions with out-degree 0 are called *terminal* positions.

A subgraph  $G = (V, E)$  of a graph  $G' = (V', E')$  is called *fully-extended* if  $\forall u \in V, (u, v) \in E' \implies (v \in V) \wedge ((u, v) \in E)$ . Retrograde analysis cannot apply to the whole game graph of Chinese chess on a physical computer, because the graph is too big. Therefore, the algorithm is applied to a fully-extended subgraph. In practice, this subgraph is usually split into multiple endgame databases according to the numbers of different pieces remaining on the board.

In this paper, the *60-move-rule* is ignored. We use a boolean function  $check: E \rightarrow \{\mathbf{true}, \mathbf{false}\}$  to indicate whether or not a given move is a checking move. The rules of non-mutual checking and mutual checking indefinitely are abstracted as follows.

**Definition 1** *In an infinite sequence of moves  $(v_0, v_1), (v_1, v_2), \text{ etc.}, \text{ if}$*

1.  $\forall \text{ even } i \geq 0, check((v_i, v_{i+1})) = \mathbf{true}, \text{ and}$
2.  $\forall n \geq 0, \exists \text{ odd } j > n, \text{ such that } check((v_j, v_{j+1})) = \mathbf{false},$

then the first mover loses the game because of the rule of non-mutual checking indefinitely. In addition, the game results in a draw because of mutual checking indefinitely if,

- $\forall i \geq 0, \text{check}((v_i, v_{i+1})) = \mathbf{true}$ .

Recall that chasing moves can be either allowed or forbidden. In some cases, we cannot tell whether a chasing move is allowed or forbidden without inspecting the other moves in the move sequence. We call the relevant rules *path-dependent*. To verify the completeness of the endgame databases, the problems caused by the path-dependent rules can be solved via the rule-tolerant approach in practical cases (Fang, 2004). In Asian rule set, the only path-dependent rule is (Association, 1999, page 103, rule 32): it is allowed to endlessly chase one piece on even moves and chase another on odd moves. To comply with this rule, we define boolean function  $\text{chase}: E \times P \rightarrow \{\mathbf{true}, \mathbf{false}\}$ , where  $P$  is the set of Chinese chess pieces.  $\text{chase}((u, v), p)$  indicates whether or not  $(u, v)$  is a forbidden move to chase  $p$ . See (Fang, 2004) for information about abstracting the chasing moves.

Three remarks are in order. First,  $P$  includes the Red King and Black King, denoted by  $RK$  and  $BK$ , i.e.,  $\{RK, BK\} \subset P$ . Therefore,  $\text{check}((u, v)) = \text{chase}((u, v), RK) \vee \text{chase}((u, v), BK)$ . In other words, a checking move is considered as a move to chase the King in this paper. Second, if  $\text{check}((u, v)) = \mathbf{true}$ , then  $\text{check}((w, v)) = \mathbf{true}$  for all  $(w, v) \in E$ . Using this property, the rule of checking indefinitely in (Fang *et al.*, 2004) is abstracted via a boolean function  $\text{inCheck}(v)$  which indicates whether the own King in the position  $u$  is in check or not. However,  $\text{chase}((u, v), p)$  generally lacks this property. So we use  $\text{check}(u, v)$  instead of  $\text{inCheck}(v)$ , and then checking indefinitely and chasing indefinitely can share the algorithms, theorems, and lemmas in many cases. Third, it is possible that a move chases multiple pieces at the same time. The rules of non-mutual chasing and mutual chasing indefinitely are abstracted as follows.

**Definition 2** *In an infinite sequence of moves  $(v_0, v_1), (v_1, v_2), \text{etc.}$ , if*

1.  $\exists p \in P - \{RK, BK\}$ , such that  $\forall \text{ even } i, \text{chase}((v_i, v_{i+1}), p) = \mathbf{true}$ , and
2.  $\forall q \in P$  and  $\forall n \geq 0, \exists \text{ odd } j > n$ , such that  $\text{chase}((v_j, v_{j+1}), q) = \mathbf{false}$ ,

then the first mover loses the game because of the rule of non-mutual chasing indefinitely. In addition, the game results in a draw because of mutual chasing indefinitely, if

- $\forall n \geq 0, \exists \text{ even } i > n$  and  $\text{odd } j > n$ , such that  $\text{check}((v_i, v_{i+1})) = \mathbf{false}$  and  $\text{check}((v_j, v_{j+1})) = \mathbf{false}$ .
- $\exists p \in P - \{RK, BK\}$ , such that  $\forall \text{ even } i \geq 0, \text{chase}((v_i, v_{i+1}), p) = \mathbf{true}$ .
- $\exists q \in P - \{RK, BK\}$ , such that  $\forall \text{ odd } j \geq 0, \text{chase}((v_j, v_{j+1}), q) = \mathbf{true}$ .

Since a checking move is considered as a move to chase the King in this paper, condition (2) in Definition 2 has excluded the case that the second mover checks the first mover indefinitely.

### 3. MOVE PATTERNS OF CHECKING/CHASING INDEFINITELY

To build complete endgame databases, we assume that both sides play perfectly and need to foresee whether or not a move sequence in Definition 1 or 2 is formed. For this purpose, this section introduces various move patterns of checking/chasing indefinitely.

#### 3.1 Basics

Retrograde analysis is applied to a *game graph*  $G = (V, E)$  which is directed, bipartite, and possibly cyclic. With the assumption that both players play flawlessly, a position is called a win/draw/loss if the next mover will win/draw/lose the game at the end, respectively.

**Definition 3** A win-draw-loss-unknown database of a directed, bipartite, and possibly cyclic game graph  $G = (V, E)$  is a function,  $DB : V \longrightarrow \{\mathbf{win}, \mathbf{draw}, \mathbf{loss}, \mathbf{unknown}\}$ . Each non-terminal position  $u \in V$  satisfies the following conditions.

1. If  $DB(u) = \mathbf{win}$ , then  $\exists(u, v) \in E$  such that  $DB(v) = \mathbf{loss}$ .
2. If  $DB(u) = \mathbf{loss}$ , then  $\forall(u, v) \in E$ ,  $DB(v) = \mathbf{win}$ .
3. If  $DB(u) = \mathbf{draw}$ , then  $\exists(u, v) \in E$  such that  $DB(v) = \mathbf{draw}$ , and  $\forall(u, v) \in E$ ,  $(DB(v) = \mathbf{draw}) \vee (DB(v) = \mathbf{win})$ .

A win-draw-loss database  $DB$  is a win-draw-loss-unknown database satisfying  $DB(u) \neq \mathbf{unknown}$  for all  $u \in V$ . A win-draw-loss-unknown database  $DB$  is called *fully-propagated*, if  $\forall u \in V$  with  $DB(u) = \mathbf{unknown}$ ,  $\exists(u, v) \in E$  such that  $DB(v) = \mathbf{unknown}$ , and  $\forall(u, v) \in E$ ,  $DB(v) \neq \mathbf{loss}$ . A win-draw-loss-unknown database  $DB$  is called *semi-fully-propagated*, if  $\forall u \in V$  with  $DB(u) = \mathbf{unknown}$ ,  $\exists(u, v) \in E$  such that  $DB(v) = \mathbf{unknown}$  or  $DB(v) = \mathbf{draw}$ , and  $\forall(u, v) \in E$ ,  $DB(v) \neq \mathbf{loss}$ .

A fully-propagated database guarantees that all positions are fully propagated. A semi-fully-propagated database guarantees that all win and loss positions are fully propagated, but draws may not be fully propagated. The algorithms in (Fang *et al.*, 2004) do not have draw positions prior the final phase, so being semi-fully-propagated is equivalent to being fully-propagated.

The classical retrograde algorithm for constructing the win-draw-loss endgame databases consists of three phases: initialization, propagation, and the final phase. In the initialization phase, the win and loss terminal positions (the *seeds*) are assigned as being **win** and **loss**, respectively. They are checkmate or stalemate positions in Chinese chess<sup>7</sup>. In the propagation phase, these values are propagated to the other positions, called *propagated* positions. The final phase is to deal with the remaining unknown positions. In chess, when no propagation can be done, the unknown positions are marked as draws. See (Fang *et al.*, 2004, Algorithm 1) for an example pseudo-code of a classical retrograde algorithm.

The following notation is used in this paper. The union and intersection of two graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are denoted as

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2) \text{ and } G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2),$$

respectively. If  $G_1$  is a *subgraph* of  $G_2$  (i.e.,  $V_1 \subseteq V_2$  and  $E_1 \subseteq E_2$ ), it is denoted by  $G_1 \subseteq G_2$ . If  $G_1$  is a *proper subgraph* of  $G_2$  (i.e.,  $G_1 \subseteq G_2$  but  $G_1 \neq G_2$ ), it is denoted by  $G_1 \subset G_2$ . Given  $G$  as a subgraph of Chinese-chess game graph, we use  $V(G)$  to denote the vertex set of  $G$ , and  $V_R(G)/V_B(G)$  to denote the sets of vertices in which Red/Black is the next mover, respectively. Note that  $V_R(G) \cap V_B(G) = \emptyset$  and  $V_R(G) \cup V_B(G) = V(G)$ . Since  $G$  is bipartite, for any  $(u, v) \in E$ ,  $u \in V_R(G)$  if and only if  $v \in V_B(G)$ . A *null* graph is also denoted by  $\emptyset$ , though formally it is  $(\emptyset, \emptyset)$ . Hereafter, all the graphs in this paper are subgraphs of Chinese-chess game graph, unless otherwise noted.

### 3.2 Checking/Chasing Indefinitely

For convenience of discussion, we assume that one player is *attacking*, whereas the other is *defending*. The attacker tries to win the game by forcing the defender to check or chase indefinitely. We begin with Definition 4, the move pattern of checking/chasing indefinitely.

**Definition 4** Given a win-draw-loss-unknown database  $DB$  for graph  $G = (V, E)$ , a move pattern of checking/chasing indefinitely is a subgraph of  $G$ , denoted by  $G^* = (V^*, E^*)$ .  $G^*$  satisfies the following conditions, where Red is the attacker and  $p$  is the Red piece being chased.

1.  $\forall u \in V^*$ ,  $DB(u) = \mathbf{unknown}$ .
2.  $\forall(u, v) \in E^*$  with  $u \in V_B(G^*)$ ,  $\mathit{chase}((u, v), p) = \mathbf{true}$ .
3.  $\forall u \in V_B(G^*)$ ,  $((u, v) \in E) \implies (DB(v) = \mathbf{unknown}) \vee (DB(v) = \mathbf{win})$ .

<sup>7</sup>In Chinese chess, the player being stalemated loses the game, whereas the game ends in a draw in Western chess.

4.  $\forall u \in V_B(G^*), ((u, v) \in E) \wedge (DB(v) = \mathbf{unknown}) \implies ((u, v) \in E^*)$ .
5.  $\forall u \in V^*, \exists(u, v) \in E^*$ , i.e., out-degree is at least 1.

If  $p = RK$ , then  $G^*$  is called a move pattern of checking indefinitely<sup>8</sup>. If  $p$  is any other Red piece, then  $G^*$  is called a pattern of chasing indefinitely. Replacing  $p$  by a Black piece and  $V_B(G^*)$  by  $V_R(G^*)$  in the conditions, we obtain the definition of a move pattern of checking/chasing indefinitely with Black as the attacker. The set of all move patterns of checking/chasing  $p$  indefinitely is denoted by  $G^*(DB, G, p)$ .

Condition (1) is because we are concerned with only the unknown positions. Condition (2) ensures the defender plays checking/chasing moves all the time inside the pattern. Conditions (3) and (4) make the defender unable to quit the pattern in  $G$  without losing the game. Condition (5) keeps the pattern indefinite. Assuming  $G$  is fully-extended, the attacker can always force the defender to check or chase indefinitely (i.e., condition (1) in Definition 1 or Definition 2 is always satisfied). However, we ignore the effect of mutual check/chasing indefinitely (i.e., condition (2) in Definition 1 or Definition 2 may not be satisfied), which is discussed in Subsections 3.4 and 3.5.

Note that Definition 4 is *relaxed* to allow  $G$  being a general subgraph of Chinese chess, whereas being fully-extended is required in (Fang *et al.*, 2004) and (Fang, 2004). The attacker in this pattern can win the game if,

1. The given graph  $G$  is a fully-extended subgraph of Chinese-chess game graph.
2. The win/draw/loss information in  $DB$  is correct.
3. The rules of mutual checking/chasing indefinitely do not take effect (i.e., condition (2) in Definition 1 for checking indefinitely or Definition 2 for chasing indefinitely is satisfied).

In this paper, a move pattern is a subgraph of the Chinese-chess game graph. A move pattern is called *empty* if it is a null graph. A move pattern  $\overline{G}$  of a certain kind is called *maximum*, if for any move pattern  $G$  of that kind,  $G \subseteq \overline{G}$ . It is clear that if a maximum move pattern  $\overline{G}$  of a certain kind exists, it is unique<sup>9</sup>.

**Lemma 1** Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a piece  $p$ , the move patterns of chasing the piece  $p$  indefinitely are closed under the union operation (i.e.,  $\forall G_1, G_2 \in G^*(DB, G, p)$ ,  $G_1 \cup G_2 \in G^*(DB, G, p)$ ).

**Proof** The result is obtained directly from verifying all the conditions in Definition 4.  $\square$

**Lemma 2** Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a chased piece  $p$ , there exists an unique maximum move pattern of chasing  $p$  indefinitely, denoted by  $\overline{G^*}(DB, G, p)$ . In particular, if  $p = RK$  or  $p = BK$ ,  $\overline{G^*}(DB, G, p)$  is called a maximum move pattern of checking indefinitely.

**Proof** The proof is analogous to that of (Fang, 2004, Theorem 1). The game graph of Chinese chess is finite, so the number of move patterns of checking/chasing indefinitely is finite. We take the union of all the move patterns of chasing  $p$  indefinitely in  $G^*(DB, G, p)$ , denoted by  $\overline{G^*}(DB, G, p)$ . By Lemma 1,  $\overline{G^*}(DB, G, p)$  is a move pattern of chasing  $p$  indefinitely. It is clearly that  $\overline{G^*}(DB, G, p)$  is maximum and unique. Note that  $\overline{G^*}(DB, G, p)$  can be a null graph.  $\square$

**Theorem 1** The maximum move pattern of checking/chasing indefinitely  $\overline{G^*}(DB, G, p)$  is an induced subgraph of  $G$ , where  $DB$  is a win-draw-loss-unknown endgame database of graph  $G$  and  $p$  is the chased piece.

**Proof** The proof is analogous to that of (Fang, 2004, Lemma 3). Denote  $G = (V, E)$  and  $\overline{G^*}(DB, G, p) = (V^*, E^*)$ . Given  $u, v \in V^*$  with  $(u, v) \in E$ , if the next mover of  $u$  is the defender,  $(u, v) \in E^*$  because of conditions (1) and (4) in Definition 4. If the next mover of  $u$  is the attacker, then adding  $(u, v)$  to  $E^*$  still satisfies Definition 4 and the move pattern  $\overline{G^*}(DB, G, p) = (V^*, E^*)$  is maximum, so  $(u, v) \in E^*$ . Therefore,  $\overline{G^*}(DB, G, p)$  is an induced subgraph of  $G$ .  $\square$

<sup>8</sup>In (Fang *et al.*, 2004), the corresponding move pattern is called of direct checking indefinitely. In this paper, the word "direct" is omitted for simplicity of terms.

<sup>9</sup>If we are given two maximum move patterns of a certain kind  $\overline{G_1}$  and  $\overline{G_2}$ , then  $\overline{G_1} \subseteq \overline{G_2}$  and  $\overline{G_2} \subseteq \overline{G_1}$ . Therefore,  $\overline{G_1} = \overline{G_2}$ .

The algorithm to compute  $\overline{G^*}(DB, G, p)$  consists of the initialization phase and the propagation phase. It is analogous to the algorithm for computing the maximum suspicious move pattern of special rules in (Fang, 2004). The key property needed is Theorem 1:  $\overline{G^*}(DB, G, p)$  is an induced subgraph of  $G$ . In the initialization phase,  $W$  and  $L$  are initialized as sets of win and loss candidates for the vertices in  $\overline{G^*}(DB, G, p)$ , so that the graph induced by  $W \cup L$  is a supergraph of  $\overline{G^*}(DB, G, p)$ . In the pruning phase, unqualified candidates are pruned. The process continues until all candidates satisfy Definition 4. Therefore, the graph induced by the resulting  $W \cup L$  is  $\overline{G^*}(DB, G, p)$ . The pseudo-code is given in Algorithm 1. Note that the children counting strategy can be applied to improve efficiency, but is excluded here for neat pseudo-code.

---

**Algorithm 1** Computing the Maximum Move Pattern of Checking/Chasing Indefinitely
 

---

```

function  $\overline{G^*}(DB$  as a database,  $G = (V, E)$  as a graph,  $p$  as a piece of the attacker) as a graph
   $W \leftarrow \emptyset, L \leftarrow \emptyset$  ▷ Initialization Phase
  for all  $u \in V$  with  $DB(u) = \mathbf{unknown}$  and the next mover of  $u$  is the defender do
    if  $\forall (u, v) \in E, (DB(v) = \mathbf{unknown}$  or  $\mathbf{win}) \wedge (DB(v) = \mathbf{unknown} \implies \mathit{chase}((u, v), p) =$ 
      true) then
       $L \leftarrow L \cup \{u\}$ 
      for all  $(w, u) \in E$  with  $DB(w) = \mathbf{unknown}$  do
         $W \leftarrow W \cup \{w\}$ 
      end for
    end if
  end for
  repeat ▷ Pruning Phase
    for all  $u \in L$  do ▷ Pruning unqualified loss candidates.
      if  $\exists (u, v) \in E$  such that  $(DB(v) = \mathbf{unknown}) \wedge (v \notin W)$  then
         $L \leftarrow L - \{u\}$ 
      end if
    end for
    for all  $v \in W$  do ▷ Pruning unqualified win candidates.
      if  $\forall (v, u) \in E, u \notin L$  then
         $W \leftarrow W - \{v\}$ 
      end if
    end for
  until No more pruning is possible.
  Return the graph induced by  $W \cup L$ .
end function

```

---

### 3.3 Mutual Checking/Chasing Indefinitely

Mutual checking indefinitely is unlikely to happen in the endgames of Chinese chess. In (Fang *et al.*, 2004), an algorithm is given to verify whether or not a given database is contaminated by the rule of mutual checking indefinitely for practical cases. Mutual chasing indefinitely has similar properties to those of mutual checking indefinitely. There are three types of problems caused by the rule of mutual checking indefinitely (Fang *et al.*, 2004). The cases of mutual chasing indefinitely are similar. The three types of problems are listed below.

1. Both players are intentionally forming the move sequence of mutual checking/chasing indefinitely regardless of the result of the game.
2. Both players are forced to mutually check/chase each other indefinitely to avoid losing the game (e.g., the example in Figure 1(a)).
3. In a move pattern of checking/chasing indefinitely in Definition 4, the attacker may not be able to force the game staying in this pattern without checking/chasing the defender indefinitely at the same time (i.e., condition (2) in Definitions 1 and 2 may not be satisfied).

Type one problems are not of concern, since we assume both players play flawlessly. The discussion in this section takes care of the type two problems. Subsections 3.4 and 3.5 give a solution to the type three problems.

**Definition 5** Given a win-draw-loss-unknown database  $DB$  for graph  $G = (V, E)$ , a move pattern of mutual checking/chasing indefinitely is a subgraph of  $G$ , denoted by  $G^* = (V^*, E^*)$ . We use  $p$  to denote the Black piece chased by Red, and  $q$  to denote the Red piece chased by Black.  $G^*$  satisfies the following conditions.

1.  $\forall u \in V^*, DB(u) = \mathbf{unknown}$ .
2.  $\forall (u, v) \in E^*$  with  $u \in V_R(G^*)$ ,  $\mathit{chase}((u, v), p) = \mathbf{true}$ .
3.  $\forall (v, u) \in E^*$  with  $v \in V_B(G^*)$ ,  $\mathit{chase}((v, u), q) = \mathbf{true}$ .
4.  $\forall u \in V^*, ((u, v) \in E) \implies (DB(v) = \mathbf{unknown}) \vee (DB(v) = \mathbf{win})$ .
5.  $\forall u \in V^*, ((u, v) \in E) \wedge (DB(v) = \mathbf{unknown}) \implies ((u, v) \in E^*)$ .
6.  $\forall u \in V^*, \exists (u, v) \in E^*$ , i.e., out-degree is at least 1.

The set of all move patterns of mutual checking/chasing indefinitely is denoted by  $G^*(DB, G, p, q)$ . It is also written as  $\overline{G^*}(DB, G, p, q)$  for flexibility.

Condition (1) is because we are concerned with only the unknown positions. Conditions (2) and (3) ensure both players play checking/chasing moves all the time inside the pattern. Conditions (4) and (5) make both players unable to quit the pattern in  $G$  without losing the game. Condition (6) keeps the pattern indefinite. Both players in this pattern can force each other to check/chase indefinitely. If  $p, q$  are the Kings of the different sides, any move sequence in this pattern satisfies the condition for mutual checking indefinitely in Definition 1. If both  $p$  and  $q$  are not Kings, we need also consider the first condition for mutual chasing indefinitely in Definition 2.

**Lemma 3** Given a win-draw-loss-unknown  $DB$  for graph  $G$  and two chased pieces  $p, q$  of different sides, any move pattern of mutual checking/chasing indefinitely is an induced subgraph of  $G$ .

**Proof** The result follows directly from conditions (1), (4) and (5) in Definition 5.  $\square$

**Lemma 4** Given a win-draw-loss-unknown  $DB$  for graph  $G$  and two chased pieces  $p, q$  of different sides, the move patterns of mutual checking/chasing indefinitely are closed under the union operation (i.e.,  $\forall G_1, G_2 \in G^*(DB, G, p, q), G_1 \cup G_2 \in G^*(DB, G, p, q)$ ).

**Proof** The result follows directly from verifying all the conditions in Definition 4.  $\square$

**Lemma 5** Suppose we are given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and two chased pieces  $p, q$  of different sides. There exists a unique maximum move pattern of checking/chasing indefinitely, denoted by  $\overline{G^*}(DB, G, p, q)$ .

**Proof** The proof is analogous to that of Lemma 2. The number of move patterns of mutual checking/chasing indefinitely is finite. Take the union of all these patterns of mutually chasing  $p$  and  $q$ , denoted by  $\overline{G^*}(DB, G, p, q)$ . By Lemma 4,  $\overline{G^*}(DB, G, p, q)$  is also a move pattern of mutually chasing  $p$  and  $q$ , which is maximum.  $\square$

By Lemma 3 and condition (6) in Definition 5, the maximum move pattern of mutual checking/chasing indefinitely  $\overline{G^*}(DB, G, p, q)$  consists of one or more separated connected components<sup>10</sup>. Each component is a move pattern of mutual checking/chasing indefinitely in  $G^*(DB, G, p, q)$ . The algorithm to compute  $\overline{G^*}(DB, G, p, q)$  consists of two phases: initialization phase and pruning phase. The key property needed is that the maximum move pattern of mutual checking/chasing indefinitely  $\overline{G^*}(DB, G, p, q)$  is an induced subgraph of  $G$ . In the initialization phase, candidates are initialized as a superset of the vertex set of  $\overline{G^*}(DB, G, p, q)$ . In the pruning phase, unqualified candidates are pruned until all candidates satisfy Definition 5. The graph induced by the remaining candidates is  $\overline{G^*}(DB, G, p, q)$ . The pseudo-code is given in Algorithm 2.

<sup>10</sup>Here we treat the graph as *undirected*. Two vertices are connected if there is a path between them. A connected component is a graph with all vertices in them connected.



**Algorithm 2** Computing the Maximum Move Pattern of Mutual Checking/Chasing Indefinitely

---

```

function  $\overline{G^*}(DB$  as a database,  $G = (V, E)$  as a graph,  $p$  as a Red piece,  $q$  as a black piece) as a graph
   $V_1 \leftarrow \emptyset, V_2 \leftarrow \emptyset$  ▷ Initialization Phase
  for all  $u \in V_B(G)$  with  $DB(u) = \text{unknown}$  do
    if  $\forall (u, v) \in E, (DB(v) = \text{unknown or win}) \wedge (DB(v) = \text{unknown} \implies \text{chase}((u, v), p) = \text{true})$  then
       $V_1 \leftarrow V_1 \cup \{u\}$ 
    end if
  end for
  for all  $v \in V_R(G)$  with  $DB(v) = \text{unknown}$  do
    if  $\forall (v, u) \in E, (DB(u) = \text{unknown or win}) \wedge (DB(u) = \text{unknown} \implies \text{chase}((v, u), q) = \text{true})$  then
       $V_2 \leftarrow V_2 \cup \{v\}$ 
    end if
  end for
  repeat ▷ Pruning Phase
    for all  $u \in V_1$  do ▷ Pruning unqualified Black-to-move candidates.
      if  $\exists (u, v) \in E$  such that  $(DB(v) = \text{unknown}) \wedge (v \notin V_2)$  then
         $V_1 \leftarrow V_1 - \{u\}$ 
      end if
    end for
    for all  $v \in V_2$  do ▷ Pruning unqualified Red-to-move candidates.
      if  $\exists (v, u) \in E$  such that  $(DB(u) = \text{unknown}) \wedge (u \notin V_1)$  then
         $V_2 \leftarrow V_2 - \{v\}$ 
      end if
    end for
  until No more pruning is possible.
  Return the graph induced by  $V_1 \cup V_2$ .
end function

```

---

In practice, the subgraph  $G$  of the Chinese chess game graph is fully-extended, the terminal win and loss positions are correctly marked, and the win-draw-loss-unknown database  $DB$  is semi-fully-propagated. The positions in  $\overline{G^*}(DB, G, BK, RK)$  are of mutual checking indefinitely, so they can be safely marked as draws. However,  $\overline{G^*}(DB, G, BK, RK)$  is usually empty in the current practical endgame databases. The only practical endgame databases with non-empty  $\overline{G^*}(DB, G, BK, RK)$  noted so far are KRCGKRC and KRCKCPG. For  $p, q \notin \{BK, RK\}$ , the positions in  $\overline{G^*}(DB, G, p, q)$  are of mutual chasing indefinitely, since both players can force each other to chase all the time. However, we need to make sure both players cannot force each other to check indefinitely inside  $\overline{G^*}(DB, G, p, q)$ . In other words, we can safely declare the positions in  $\overline{G^*}(DB, G, p, q)$  as draws, if  $\overline{G^*}(DB, \overline{G^*}(DB, G, p, q), RK) = \emptyset$  and  $\overline{G^*}(DB, \overline{G^*}(DB, G, p, q), BK) = \emptyset$ . In practice,  $\overline{G^*}(DB, G, p, q)$  is usually empty. For example, in the selected 50 endgame databases in (Fang *et al.*, 2004; Fang, 2004),  $\overline{G^*}(DB, G, p, q) = \emptyset$  for  $p, q \notin \{BK, RK\}$ .

### 3.4 Non-mutual Checking/Chasing Indefinitely

Now we consider the type three problems caused by mutual checking/chasing indefinitely. Our goal is to find a move pattern like the maximum move pattern of checking/chasing indefinitely in Lemma 2, and the condition (2) in Definitions 1 and 2 is also guaranteed. We begin with Definition 6.

**Definition 6** Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and a piece  $p$  of the attacker and another piece  $q$  of the defender, in graph  $G_1 \in \overline{G^*}(DB, G, p)$  it is said that the attacker is free from being forced to chase  $q$  indefinitely, if  $G^*(DB, G_1, q) = \emptyset$ . This  $G_1$  is called a pattern of non-mutual checking/chasing indefinitely. We denote the set of all these graphs by  $G_{\dagger}^*(DB, G, p, q)$ .

In pattern  $G_1$  of Definition 6 satisfies that the attacker can force the defender to chase  $p$  all the time, and the defender cannot force the attacker to chase  $q$  indefinitely inside  $G_1$ . For  $p = RK$  and  $q = BK$ , it is a move

pattern of non-mutual checking indefinitely. The attacker can always form a move sequence satisfying both conditions (1) and (2) in Definition 1, unless the defender quits the pattern and loses the game.

**Lemma 6** *Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$ , a piece  $p$  of the attacker and a piece  $q$  of the defender, the move patterns of non-mutual checking/chasing indefinitely are closed under the union operation (i.e.,  $\forall G_1, G_2 \in G_{\dagger}^*(DB, G, p, q)$ ,  $G_1 \cup G_2 \in G_{\dagger}^*(DB, G, p, q)$ ).*

**Proof** By Definition 6,  $G_1, G_2 \in G^*(DB, G, p)$  for any given  $G_1, G_2 \in G_{\dagger}^*(DB, G, p, q)$ . By Lemma 1,  $G_1 \cup G_2 \in G^*(DB, G, p)$ . Let  $G_0 = \overline{G^*}(DB, G_1 \cup G_2, q)$ . The rest of the proof is to show  $G_0 = \emptyset$  by contradiction. Suppose  $G_0 \neq \emptyset$ . Since  $G_0 \subseteq G_1 \cup G_2$ , either  $G_0 \cap G_1 \neq \emptyset$  or  $G_0 \cap G_2 \neq \emptyset$ . Without loss of generality, we assume  $G_0 \cap G_1 \neq \emptyset$ . Now we claim  $G_0 \cap G_1 \in G^*(DB, G_1, q)$ . Since  $G_0 = \overline{G^*}(DB, G_1 \cup G_2, q)$ ,  $G_0 \cap G_1$  satisfies the first four conditions in Definition 4 for being a move pattern in  $G^*(DB, G_1, q)$ . The investigation of the last condition is as follows. Denote  $G = (V, E)$ ,  $G_0 = (V_0, E_0)$ ,  $G_1 = (V_1, E_1)$  and  $G_0 \cap G_1 = (\overline{V}, \overline{E})$ . Given a vertex  $u \in \overline{V}$ , if the next mover in  $u \in V_1$  is the defender in  $G_1$ , then  $\forall (u, v) \in E$  with  $DB(v) = \mathbf{unknown}$ ,  $(u, v) \in E_1$ . Since  $u \in V_0$ , there exists  $(u, w) \in E_0$  with  $DB(w) = \mathbf{unknown}$ . Note that  $(u, w) \in E_1$  and therefore  $(u, w) \in \overline{E}$ . If the next mover in  $u$  is the attacker in  $G_1$  (defender in  $G_0$ ), for all edges  $(u, v)$  in the edge set of  $G_1 \cup G_2$ ,  $(u, v) \in E_0$ , since  $G_0 = \overline{G^*}(DB, G_1 \cup G_2, q)$ . There exists  $(u, w) \in E_1$  since  $u \in V_1$ . Note that  $(u, w) \in E_0$  and therefore  $(u, w) \in \overline{E}$ . We conclude that for any  $u \in \overline{V}$ , there exists  $(u, w) \in \overline{E}$ , no matter whether the next mover in  $u$  is the attacker or the defender.  $G_0 \cap G_1$  also satisfies the last condition in Definition 4 for being a move pattern in  $G^*(DB, G_1, q)$ . Therefore,  $G_0 \cap G_1 \in G^*(DB, G_1, q)$ . However,  $G^*(DB, G_1, q) = \emptyset$  since  $G_1 \in G_{\dagger}^*(DB, G, p, q)$ . This is a contradiction. Therefore,  $G_0 = \overline{G^*}(DB, G_1 \cup G_2, q) = \emptyset$ , so that  $G_1 \cup G_2 \in G_{\dagger}^*(DB, G, p, q)$ .

**Lemma 7** *Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a chased piece  $p$  of the attacker and another chased piece  $q$  of the defender, there exists a unique maximum move pattern of non-mutual checking/chasing indefinitely, denoted by  $\overline{G_{\dagger}^*}(DB, G, p, q)$ . In particular,  $\overline{G_{\dagger}^*}(DB, G, RK, BK)$  and  $\overline{G_{\dagger}^*}(DB, G, BK, RK)$  are called maximum move patterns of mutual checking indefinitely.*

**Proof** The proof is analogous to that of Lemma 2. The game graph of Chinese chess is finite, so the number of move patterns of non-mutual checking/chasing indefinitely is finite. We take the union of all the move patterns of mutual checking/chasing indefinitely in  $G_{\dagger}^*(DB, G, p, q)$ , denoted by  $\overline{G_{\dagger}^*}(DB, G, p, q)$ . By Lemma 6,  $\overline{G_{\dagger}^*}(DB, G, p, q) \in G_{\dagger}^*(DB, G, p, q)$ . It is clear that  $\overline{G_{\dagger}^*}(DB, G, p, q)$  is maximum and unique.

**Theorem 2**  *$\overline{G_{\dagger}^*}(DB, G, p, q)$  is an induced subgraph of  $G$ , where  $DB$  is a win-draw-loss-unknown endgame database for graph  $G$ , and  $p, q$  are the chased pieces of different sides.*

**Proof** Denote  $G = (V, E)$  and  $\overline{G_{\dagger}^*}(DB, G, p, q) = (V^*, E^*)$ . Note that  $\overline{G_{\dagger}^*}(DB, G, p, q) \in G^*(DB, G, p)$ . Given  $u, v \in V^*$  with  $(u, v) \in E$  with the next mover in  $u$  being the defender. Then  $(u, v) \in E^*$  because of conditions (1) and (4) in Definition 4. Suppose there are  $v, u \in V^*$  with  $(v, u) \in E$  but  $(v, u) \notin E^*$ , where the next mover in  $v$  is the attacker. Denote the move pattern  $\overline{G_{\dagger}^*}(DB, G, p, q)$  after adding  $(v, u)$  to  $E^*$  by  $G_1$ . Now we claim that  $G_1 \in G_{\dagger}^*(DB, G, p, q)$ . First,  $G_1 \in G^*(DB, G, p)$ , since adding a move of the attacker still satisfies Definition 4. Suppose  $\overline{G^*}(DB, G_1, q) \neq \emptyset$ ,  $\overline{G^*}(DB, G_1, q)$  contains the added edge  $(v, u)$ , since  $\overline{G^*}(DB, \overline{G_{\dagger}^*}(DB, G, p, q), q) = \emptyset$ . The vertex  $v$  must have another out-going edge denoted by  $(v, w)$  other than  $(v, u)$  in  $\overline{G_{\dagger}^*}(DB, G, p, q)$ , because  $\overline{G_{\dagger}^*}(DB, G, p, q) \in G^*(DB, G, p)$  satisfies condition (5) in Definition 4.  $\overline{G^*}(DB, G_1, q)$  contains the edge  $(v, w)$  because of conditions (1) and (4) in Definition 4. Denote the move pattern  $\overline{G^*}(DB, G_1, q)$  after removing the edge  $(u, v)$  by  $G_2$ . Then  $G_2 \in G^*(DB, \overline{G_{\dagger}^*}(DB, G, p, q), q)$ .  $G_2$  is non-empty since it contains  $(v, w)$ . However,  $G^*(DB, \overline{G_{\dagger}^*}(DB, G, p, q), q) = \emptyset$  by Definition 6. A contradiction. Therefore,  $\overline{G^*}(DB, G_1, q) = \emptyset$  and  $G_1 \in G_{\dagger}^*(DB, G, p, q)$ . However,  $\overline{G_{\dagger}^*}(DB, G, p, q)$  is maximum but  $\overline{G_{\dagger}^*}(DB, G, p, q) \subset G_1$ . We conclude that such a move  $(v, u)$  does not exist. As a result,  $\overline{G_{\dagger}^*}(DB, G, p, q)$  is an induced subgraph of  $G$ .  $\square$

The algorithm to compute  $\overline{G_{\dagger}^*}(DB, G, p, q)$  consists of the initialization phase and the pruning phase. The sketch is described as follows. In the initialization phase, an induced subgraph of  $G$  is initialized as a supergraph of  $\overline{G_{\dagger}^*}(DB, G, p, q)$ , e.g.,  $\overline{G^*}(DB, G, p)$ . In the pruning phase, all the unqualified vertices are pruned, until the graph induced by the remaining vertices satisfies Definition 6. By Theorem 2, the resulting graph is  $\overline{G_{\dagger}^*}(DB, G, p, q)$ .

How to prune the unqualified vertices is described as follows. First, the vertices in the move pattern  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  do not satisfy Definition 6, where the defender can force the attacker to chase  $q$  indefinitely inside  $\overline{G^*}(DB, G, p)$ . For non-mutual checking indefinitely, these positions do not satisfy condition (2) in Definition 1. For non-mutual chasing indefinitely, these positions do not satisfy condition (2) in Definition 2. Therefore, they may be draws. We call them *potential draws*, whereas the other attacker-to-move and defender-to-move positions in  $\overline{G^*}(DB, G, p)$  are potential wins and potential losses, respectively. These potential draws in  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  can be propagated back to the predecessors. If a potential loss has a child as a potential draw, it is updated to be a potential draw. If a potential win has no child as a potential loss, it is updated to be a potential draw. The pseudo-code is given in Algorithm 3.

---

**Algorithm 3** Algorithm to Exclude Mutual Checking/Chasing Indefinitely
 

---

**Require:**  $G_1 \in G^*(DB, G, p)$  and  $G_1$  is an induced subgraph of  $G$ .

**Ensure:**  $G^\dagger(DB, G_1, q) \in G^*(DB, G, p)$  and  $G^\dagger(DB, G_1, q)$  is an induced subgraph of  $G$ .

**function**  $G^\dagger(DB$  as a database,  $G_1$  as a graph,  $q$  as a piece of defender) as a graph

Denote the vertex set of  $G_1$  by  $W \cup L$ . The next mover in  $W/L$  is the attacker/defender, respectively.  
 Compute the vertex set  $W^* \cup L^*$  of  $\overline{G^*}(DB, G_1, q)$  by Algorithm 1. ▷ Initialization Phase

$L \leftarrow L - W^*$

$W \leftarrow W - L^*$

**while**  $W^* \cup L^* \neq \emptyset$  **do** ▷ Pruning Phase

**while**  $W^* \neq \emptyset$  **do** ▷ Pruning unqualified win candidates.

    Pop any  $u \in W^*$  and set  $W^* \leftarrow W^* - \{u\}$ .

**for all**  $(v, u) \in E$  with  $v \in W$  **do**

**if**  $\forall (v, w) \in E, w \notin L$  **then**

$W \leftarrow W - \{v\}; L^* \leftarrow L^* \cup \{v\}$  ▷ (\*)

**end if**

**end for**

**end while**

**while**  $L^* \neq \emptyset$  **do** ▷ Pruning unqualified loss candidates.

    Pop any  $v \in L^*$  and set  $L^* \leftarrow L^* - \{v\}$ .

**for all**  $(u, v) \in E$  with  $u \in L$  **do**

$L \leftarrow L - \{u\}; W^* \leftarrow W^* \cup \{u\}$  ▷ (\*\*)

**end for**

**end while**

**end while**

  Return the graph induced by  $W \cup L$ .

**end function**

---

Three remarks for Algorithm 3 are given as follows. First, the next mover in  $W$  and  $L^*$  is the attacker, whereas the next mover in  $L$  and  $W^*$  is the defender. Second, the defender in  $W^*$  can force the attacker also to check/chase at the same time in  $G^*(DB, G, p)$ , so the status of the game cannot yet be determined. Third, the children counting strategy can be applied to improve efficiency, but is excluded here for simplicity. Lemma 8 ensures that  $G^\dagger(DB, G_1, q)$  remains a move pattern of checking/chasing indefinitely in  $G^*(DB, G, p)$ , assuming  $G_1$  is also a move pattern of checking/chasing indefinitely in  $G^*(DB, G, p)$  and an induced graph of  $G$ , where  $p, q$  are the chased pieces of the attacker and defender, respectively.

**Lemma 8** *Given a win-draw-loss-unknown database  $DB$  for graph  $G = (V, E)$  and two pieces  $p, q$  of different sides and an induced subgraph  $G_1$  of  $G$ ,*

$$G_1 \in G^*(DB, G, p) \implies G^\dagger(DB, G_1, q) \in G^*(DB, G, p).$$

**Proof** Suppose  $G_1 \in G^*(DB, G, p)$ . Since  $G^\dagger(DB, G_1, q) \subseteq G_1$ ,  $G^\dagger(DB, G_1, q)$  satisfies Conditions (1), (2) and (3) in Definition 4. The operation (\*\*) in Algorithm 3 guarantees that Condition (4) holds. The operations (\*) and (\*\*) ensures Condition (5). Therefore,  $G^\dagger(DB, G_1, q) \in G^*(DB, G, p)$ .  $\square$

Suppose we are given a win-draw-loss-unknown database  $DB$  for graph  $G$ , and pieces  $p, q$  of different sides. To get the maximum move pattern of non-mutual checking/chasing indefinitely  $\overline{G^*}_\dagger(DB, G, p, q)$ , we first compute  $\overline{G^*}(DB, G, p)$ , then get the reduced graph  $G^\dagger(DB, \overline{G^*}(DB, G, p), q)$  after pruning some unqualified vertices. The operation is repeated until the graph cannot be trimmed any more. As a result,

the defender has no chance to force the attacker to chase  $q$  indefinitely at the same time in the pattern. Therefore, Definition 6 is satisfied and the move pattern obtained is  $\overline{G}_\dagger^*(DB, G, p, q)$ . The pseudo-code is given in Algorithm 4.

---

**Algorithm 4** Computing  $\overline{G}_\dagger^*(DB, G, p, q)$  of Non-Mutual Checking/Chasing Indefinitely

---

**function**  $\overline{G}_\dagger^*(DB$  as a database,  $G$  as a graph,  $p$  as a piece of attacker,  $q$  is a piece of the defender) as a graph  
 $G_1 \leftarrow \overline{G}^*(DB, G, p)$ , computed by Algorithm 1. ▷ Initialization Phase  
**repeat** ▷ Propagation Phase  
 $G_1 \leftarrow G^\dagger(DB, G_1, q)$ , computed by Algorithm 3.  
**until**  $G_1 = G^\dagger(DB, G_1, q)$ .  
return  $G_1$   
**end function**

---

### 3.5 Totally Non-Mutual Checking/Chasing Indefinitely

In a move pattern of chasing indefinitely, we need to consider that the defender may try to force the attacker to chase any possible piece indefinitely (i.e., condition (2) in Definition 2 needs to be satisfied). Therefore, the move pattern of *totally non-mutual checking/chasing indefinitely* is defined as follows.

**Definition 7** Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and a piece  $p$  of the attacker, in graph  $G_1 \in G^*(DB, G, p)$  it is said that the attacker is free from being forced to check/chase the defender indefinitely if  $G^*(DB, G_1, q) = \emptyset$  for all pieces of the defender  $q$  including the King. This  $G_1$  is called a pattern of *totally non-mutual checking/chasing indefinitely*. We denote the set of all these graphs by  $G_\dagger^*(DB, G, p)$ .

In the pattern  $G_1$  of Definition 7, the attacker can force the defender to chase  $p$  all the time, and the defender cannot force the attacker to chase any pieces indefinitely inside  $G_1$ . Assuming that  $p$  is not the King, the attacker in  $G_1$  can always form a move sequence satisfying both conditions (1) and (2) in Definition 2, unless the defender quits the pattern and loses the game.

The move pattern of totally non-mutual checking/chasing indefinitely has the properties similar to those of Lemmas 6 and 7 and Theorem 2 for the move pattern of non-mutual checking/chasing indefinitely, described as follows. The proofs are omitted, since they are similar to those of Lemmas 6 and 7 and Theorem 2.

**Lemma 9** Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a piece  $p$  of attacker, the move patterns of *totally non-mutual checking/chasing indefinitely* are closed under the union operation (i.e.,  $\forall G_1, G_2 \in G_\dagger^*(DB, G, p)$ ,  $G_1 \cup G_2 \in G_\dagger^*(DB, G, p)$ ).

**Lemma 10** Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a chased piece  $p$  of the attacker, there exists an unique maximum move pattern of *totally non-mutual checking/chasing indefinitely*, denoted by  $\overline{G}_\dagger^*(DB, G, p)$ .

**Theorem 3**  $\overline{G}_\dagger^*(DB, G, p)$  is an induced subgraph of  $G$ , where  $DB$  is a win-draw-loss-unknown endgame database of graph  $G$ , and  $p$  is the chased piece of the attacker.

Given a win-draw-loss-unknown endgame database  $DB$  for graph  $G$  and a piece  $p$  of the attacker, the algorithm to compute  $\overline{G}_\dagger^*(DB, G, p)$  is similar to Algorithm 4. The key property needed is Theorem 3:  $\overline{G}_\dagger^*(DB, G, p)$  is an induced subgraph of  $G$ . The algorithm consists of the initialization phase and the pruning phase. In the initialization phase,  $G_1$  is initialized as a supergraph of  $\overline{G}_\dagger^*(DB, G, p)$ . In the pruning phase, unqualified vertices in  $G_1$  are repeatedly pruned, until Definition 7 is satisfied. The pseudo-code is given in Algorithm 5.

In Algorithm 5,  $G_1$  is initialized as a supergraph of  $\overline{G}_\dagger^*(DB, G, p)$ . In the pruning phase, all the vertices pruned are unqualified no matter what the order of the defending pieces in the loop (\*) is. When no pruning is possible,  $G_1$  satisfies Definition 7 and is therefore maximum. By Lemma 10,  $\overline{G}_\dagger^*(DB, G, p)$  is unique. Therefore, no matter what the order of the defending pieces in the loop (\*) is, we obtain the same  $\overline{G}_\dagger^*(DB, G, p)$  by Algorithm 5.

**Algorithm 5** Computing  $\overline{G}_\dagger^*(DB, G, p)$  of Totally Non-Mutual Checking/Chasing Indefinitely

---

```

function  $\overline{G}_\dagger^*(DB$  as a database,  $G$  as a graph,  $p$  as a piece of attacker) as a graph
   $G_1 \leftarrow \overline{G}^*(DB, G, p)$ , computed by Algorithm 1. ▷ Initialization Phase
  repeat ▷ Pruning Phase
    for each piece  $q$  of the defender, including the King do
       $G_1 \leftarrow G^\dagger(DB, G_1, q)$ , computed by Algorithm 3. ▷ (*)
    end for
  until  $G_1$  cannot be further reduced.
  return  $G_1$ 
end function

```

---

By Definitions 6 and 7, given a database  $DB$  for graph  $G$  and two chased pieces  $p, q$  of different sides,  $\overline{G}_\dagger^*(DB, G, p, q) \in G^*(DB, G, p)$  and  $\overline{G}_\dagger^*(DB, G, p) \in G^*(DB, G, p)$ . In Algorithms 4 and 5 which invoke Algorithm 3, this property is also confirmed by iteratively applying Lemma 8.

Two final remarks of this section are given as follows. First, Algorithms 1, 2, 4 and 5 respectively to compute  $\overline{G}^*(DB, G, p)$ ,  $\overline{G}^*(DB, G, p, q)$ ,  $\overline{G}_\dagger^*(DB, G, p, q)$  and  $\overline{G}_\dagger^*(DB, G, p)$  do not require the given graph  $G$  to be fully-extended. In practice, they are applied to the fully-extended subgraphs of Chinese-chess game graph. Second, all Algorithms 1, 2, 3, 4 and 5 do not require the given database  $DB$  to be fully-propagated or semi-fully-propagated. In practice, they are applied to the semi-fully-propagated databases.

## 4. BUILDING COMPLETE WIN-DRAW-LOSS DATABASES

Given a fully-extended subgraph  $G$  of Chinese chess, the goal in this section is to build a complete win-draw-loss database  $DB$  for graph  $G$  (i.e., complying with all the rules in the Asian rule set). Section 3 has built the essential tools for this purpose. Subsection 4.1 gives an algorithm to build the complete a win-draw-loss database. Subsection 4.2 discusses how one may optionally deal with draws prior to the final phase to improve the efficiency.

### 4.1 The Algorithm

Given a game graph, a classical retrograde algorithm first determines the win and loss information of the terminal vertices, checkmate and stalemate positions, and then iteratively propagates the information back to their predecessors until no propagation is possible. In the final phase, all unknown positions are marked as draws, assuming that moving among these positions without reaching a terminal position results in a draw. In Chinese chess, however, some positions marked as draws in the final phase result in Red win or Black win because of the rules of non-mutual checking indefinitely or non-mutual chasing indefinitely. For building complete win-draw-loss endgame databases of Chinese chess, these positions have to be taken into account.

The algorithm is described as follows. After the regular initialization phase and propagation phase, we compute the maximum move patterns of non-mutual checking indefinitely. If they are empty, then the maximum move patterns of totally non-mutual chasing indefinitely are computed. After marking the win and loss positions in these patterns, the database is generally neither fully-propagated nor semi-fully-propagated. So it follows a propagation phase. After each propagation phase, the database is changed, so the new maximum move patterns of non-mutual checking or non-mutual chasing indefinitely may exist, and therefore the procedure is repeated. We say that positions propagated in different phases are of different *levels*. The process continues until all the new maximum move patterns of non-mutual checking and totally non-mutual chasing indefinitely are empty. Therefore, both players cannot force each other to violate the rules of non-mutual checking indefinitely and non-mutual chasing indefinitely. So the remaining unknown positions can be correctly marked as draws in the final phase. Since the database keeps being semi-fully-propagated after each propagation phase, the database with the draws marked in the final phase is a sound win-draw-loss database. We may ignore the positions of mutual checking indefinitely and mutual chasing indefinitely, since they are eventually correctly marked as draws in the final phase. The draws because of mutual checking indefinitely

or mutual chasing indefinitely can also be marked prior the final phase as discussed in Subsection 4.2. The pseudo-code for building a complete win-draw-loss database of a fully-extended subgraph of Chinese chess is given in Algorithm 6.

---

**Algorithm 6** Building a Complete Win-Draw-Loss Database  $DB$  for  $G = (V, E)$ 


---

```

for all  $v \in V$  do ▷ Initialization Phase
   $DB(v) \leftarrow \text{unknown}$ 
end for
 $W \leftarrow \{\text{terminal win positions}\}$ 
 $L \leftarrow \{\text{terminal loss positions}\}$ 
repeat
  repeat ▷ Propagation Phase
    for all  $u \in L$  do ▷ propagating loss positions
       $DB(u) \leftarrow \text{loss}$ 
      for all  $(v, u) \in E$  with  $DB(v) = \text{unknown}$  do
         $W \leftarrow W \cup \{v\}$ 
      end for
    end for
     $L \leftarrow \emptyset$ 
    for all  $v \in W$  do ▷ propagating win positions
       $DB(v) \leftarrow \text{win}$ 
      for all  $(u, v) \in E$  with  $DB(u) = \text{unknown}$  do
        If  $\forall (u, w) \in E, DB(w) = \text{loss}$ , then  $L \leftarrow L \cup \{u\}$ .
      end for
    end for
     $W \leftarrow \emptyset$ 
  until  $W = \emptyset$  and  $L = \emptyset$  ▷  $DB$  is semi-fully-propagated when quitting the loop.
  {In the first iteration, optionally marking terminal draw positions with propagation here.}
  {Optionally marking positions of mutual checking indefinitely with propagation here.}
  if  $\overline{G}_\dagger^*(DB, G, RK, BK) \cup \overline{G}_\dagger^*(DB, G, BK, RK) \neq \emptyset$  then ▷ non-mutual checking indefinitely
     $L \leftarrow \{\text{loss positions in } \overline{G}_\dagger^*(DB, G, RK, BK) \cup \overline{G}_\dagger^*(DB, G, BK, RK)\}$ 
  else
    {Optionally marking positions of mutual chasing indefinitely with propagation here.}
     $G_1 \leftarrow \text{union of all } \overline{G}_\dagger^*(DB, G, p)$  with  $p$  as a Red piece other than the King
     $G_2 \leftarrow \text{union of all } \overline{G}_\dagger^*(DB, G, q)$  with  $q$  as a Black piece other than the King
    if  $G_1 \cup G_2 \neq \emptyset$  then ▷ non-mutual chasing indefinitely
       $L \leftarrow \{\text{loss positions in } G_1 \cup G_2\}$ 
    else
      Break the repeat loop. ▷ (*)
    end if
  end if
  until the repeat loop breaks in (*).
  for all  $v \in V$  with  $DB(v) = \text{unknown}$  do ▷ Final Phase
     $DB(v) \leftarrow \text{draw}$ 
  end for

```

---

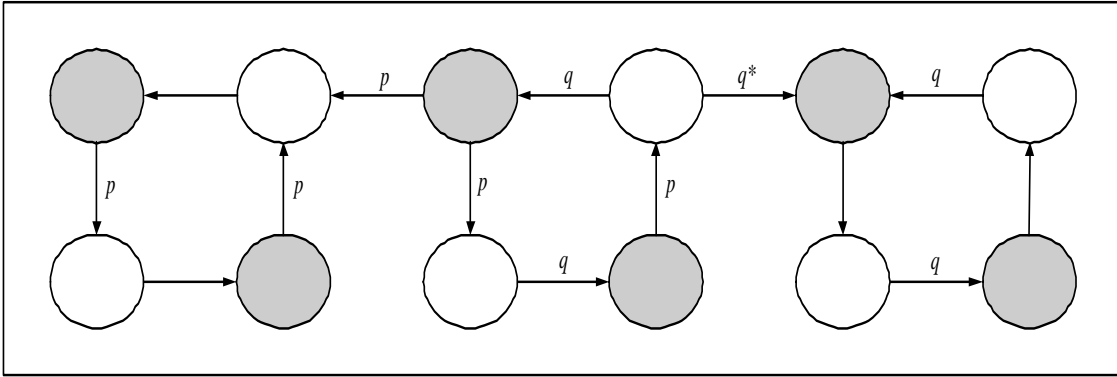
In Algorithm 6, if  $\overline{G}_\dagger^*(DB, G, RK, BK) \cap \overline{G}_\dagger^*(DB, G, BK, RK) \neq \emptyset$ , then problems occur. In other words, if a position is a win in a maximum move pattern of non-mutual checking indefinitely and also a loss in the other, then it is not only a win but also a loss and results in inconsistency. Similarly, the positions in  $G_1 \cap G_2$  in Algorithm 6 are wins in a maximum move pattern of totally non-mutual chasing indefinitely and also losses in another. If  $G_1 \cap G_2 \neq \emptyset$ , then it also results in inconsistency. Theorem 4, proved via Lemma 11, is developed to show that the inconsistency cannot happen.

**Lemma 11** Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and two chased pieces  $p, q$  of different sides,

$$\overline{G}^*(DB, G, p, q) \subseteq (\overline{G}^*(DB, G, p) \cap \overline{G}^*(DB, G, q)) \subseteq \overline{G}^*(DB, \overline{G}^*(DB, G, p), q).$$

**Proof** By Definition 5,  $\overline{G^*}(DB, G, p, q) \in G^*(DB, G, p)$ . Since  $\overline{G^*}(DB, G, p)$  is the maximum move pattern in  $G^*(DB, G, p)$ ,  $\overline{G^*}(DB, G, p, q) \subseteq \overline{G^*}(DB, G, p)$ . Similarly, we obtain  $\overline{G^*}(DB, G, p, q) \subseteq \overline{G^*}(DB, G, q)$ . Therefore,  $\overline{G^*}(DB, G, p, q) \subseteq \overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)$ .

Now we claim that  $(\overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)) \in G^*(DB, \overline{G^*}(DB, G, p), q)$ . Denote the patterns  $\overline{G^*}(DB, G, p)$  and  $\overline{G^*}(DB, G, q)$  by  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ , respectively. Since  $G_1 = \overline{G^*}(DB, G, p)$  and  $G_2 = \overline{G^*}(DB, G, q)$  satisfy the conditions in Definition 4 as being the maximum move patterns,  $G_1 \cap G_2$  satisfies all the first four conditions in Definition 4 for being a move pattern in  $G^*(DB, G_1, q)$ . Now we investigate the last condition in Definition 4. Without loss of generality, we let Red/Black be the attacker in  $G_1/G_2$ , respectively. Given  $u \in V_1 \cap V_2$ , if  $u$  is as Red to move, then  $\exists (u, v) \in E_1$  since Red is the attacker in  $G_1$ . Besides,  $(u, v) \in E_2$  since Red is the defender in  $G_2$ . Therefore,  $(u, v) \in E_1 \cap E_2$  and  $v \in V_1 \cap V_2$ . Similarly, if the next mover in  $u$  is Black, then  $\exists v \in V_1 \cap V_2$  and  $(u, v) \in E_1 \cap E_2$ . As a result, the graph does not have any terminal vertex (i.e., all vertices with out-degrees at least 1). So the last condition in Definition 4 also holds. Since  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  is the maximum move pattern in  $G^*(DB, \overline{G^*}(DB, G, p), q)$ ,  $(\overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)) \subseteq \overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$ .  $\square$



**Figure 2:** Example to illustrate the relationships of the move patterns in Lemma 11.

Figure 2 illustrates an example of the relationships of the move patterns in Lemma 11, where  $p$  indicates a move to chase the piece  $p$  (i.e.,  $check((u, v), p) = \mathbf{true}$ ) and  $q$  indicates a move to chase the piece  $q$  (i.e.,  $chase((u, v), q) = \mathbf{true}$ ). Then  $\overline{G^*}(DB, G, p)$  is the graph induced by the left eight vertices, whereas  $\overline{G^*}(DB, G, q)$  is the graph induced by the right eight vertices. Therefore, the intersection is the graph induced by the middle four vertices. Besides,  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  is also the graph induced by the middle four vertices and  $\overline{G^*}(DB, G, p, q) = \emptyset$ . We obtain

$$\overline{G^*}(DB, G, p, q) \subset (\overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)) = \overline{G^*}(DB, \overline{G^*}(DB, G, p), q).$$

If the move with  $q^*$  is not a chasing move (i.e.,  $chase((u, v), q) = \mathbf{false}$ ), then  $\overline{G^*}(DB, G, q)$  becomes the subgraph induced by the right four vertices, and therefore the intersection with  $\overline{G^*}(DB, G, p)$  is empty. The pattern  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  remains the graph induced by the middle four vertices. Therefore,

$$\overline{G^*}(DB, G, p, q) = (\overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)) \subset \overline{G^*}(DB, \overline{G^*}(DB, G, p), q).$$

**Theorem 4** Given a database  $DB$  for graph  $G$  and two pieces  $p, q$  of different sides,

$$\overline{G^*}_{\dagger}(DB, G, p, q) \cap \overline{G^*}_{\dagger}(DB, G, q, p) = \emptyset$$

and

$$\overline{G^*}_{\dagger}(DB, G, p) \cap \overline{G^*}_{\dagger}(DB, G, q) = \emptyset.$$

**Proof** Since  $\overline{G^*}_{\dagger}(DB, G, p, q) \subseteq \overline{G^*}(DB, G, p)$  and  $\overline{G^*}_{\dagger}(DB, G, q, p) \subseteq \overline{G^*}(DB, G, q)$ ,

$$(\overline{G^*}_{\dagger}(DB, G, p, q) \cap \overline{G^*}_{\dagger}(DB, G, q, p)) \subseteq (\overline{G^*}(DB, G, p) \cap \overline{G^*}(DB, G, q)).$$

In Algorithm 4 which invokes Algorithm 3,  $\overline{G^*}(DB, \overline{G^*}(DB, G, p), q)$  is removed from  $\overline{G^*}_{\dagger}(DB, G, p)$  while computing  $\overline{G^*}_{\dagger}(DB, G, p, q)$ . By Lemma 11,  $\overline{G^*}_{\dagger}(DB, G, p, q) \cap (\overline{G^*}_{\dagger}(DB, G, p) \cap \overline{G^*}_{\dagger}(DB, G, q)) =$

$\emptyset$ . Similarly,  $\overline{G}_\dagger^*(DB, G, q, p) \cap (\overline{G}_\dagger^*(DB, G, p) \cap \overline{G}_\dagger^*(DB, G, q)) = \emptyset$ . Therefore,  $\overline{G}_\dagger^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, q, p) = \emptyset$ . Since  $\overline{G}_\dagger^*(DB, G, p) \subseteq \overline{G}_\dagger^*(DB, G, p, q)$  and  $\overline{G}_\dagger^*(DB, G, q) \subseteq \overline{G}_\dagger^*(DB, G, q, p)$ ,  $\overline{G}_\dagger^*(DB, G, p) \cap \overline{G}_\dagger^*(DB, G, q) = \emptyset$ .  $\square$

By Theorem 4, it is always true that  $\overline{G}_\dagger^*(DB, G, RK, BK) \cap \overline{G}_\dagger^*(DB, G, BK, RK) = \emptyset$  and  $G_1 \cap G_2 = \emptyset$  without conflicts in Algorithm 6. Recall that if there are no draw positions in a win-draw-loss-unknown database, being fully-propagated and being semi-fully-propagated are equivalent. Therefore, if no draws are marked prior the final phase, the database is fully-propagated after each propagated phase during the construction by Algorithm 6.

## 4.2 The Draws

In Algorithm 6, after each propagation phase, the positions in the maximum move pattern of mutual checking indefinitely  $\overline{G}^*(DB, G, RK, BK)$  can be safely declared as draws. Marking these as draws in  $DB$  before the final phase does not affect the correctness of the database. However, it can improve the efficiency for computing  $\overline{G}_\dagger^*(DB, G, RK, BK)$  and  $\overline{G}_\dagger^*(DB, G, BK, RK)$  by Algorithm 4 in the later iterations. Note that the positions in  $\overline{G}^*(DB, G, RK, BK)$  are always initialized as the candidates while computing  $\overline{G}^*(DB, G, RK)$  and  $\overline{G}^*(DB, G, BK)$  by Algorithm 1 invoked by Algorithm 4 where these positions are pruned.

**Lemma 12** *Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and two pieces  $p, q$  of different sides,*

$$\overline{G}^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, p, q) = \emptyset.$$

**Proof** By Lemma 11,  $\overline{G}_\dagger^*(DB, G, p, q) \subseteq \overline{G}^*(DB, \overline{G}_\dagger^*(DB, G, p), q)$ . In Algorithm 4 which invokes Algorithm 3,  $\overline{G}_\dagger^*(DB, \overline{G}_\dagger^*(DB, G, p), q)$  is removed from  $\overline{G}^*(DB, G, p)$  while computing  $\overline{G}_\dagger^*(DB, G, p, q)$ . Therefore,  $\overline{G}_\dagger^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, p, q) = \emptyset$ .

By Theorem 4 and Lemma 12, the maximum move patterns of mutual and non-mutual checking indefinitely are all disjoint (i.e.,  $\overline{G}^*(DB, G, RK, BK)$ ,  $\overline{G}_\dagger^*(DB, G, RK, BK)$  and  $\overline{G}_\dagger^*(DB, G, BK, RK)$  are all disjoint). Therefore, the win-draw-loss information of the positions in them can be correctly marked without conflicts.

For a maximum move pattern of mutual chasing indefinitely  $\overline{G}^*(DB, G, p, q)$ , where  $p$  and  $q$  are two pieces of different sides other than the Kings, some player may be able to force the other to check indefinitely inside  $\overline{G}^*(DB, G, p, q)$ . Therefore, the positions in  $\overline{G}^*(DB, G, p, q)$  may not be safely marked as draws since the first condition for mutual chasing indefinitely in Definition 2 may not be satisfied. Nevertheless, if  $\overline{G}^*(DB, \overline{G}^*(DB, G, p, q), RK)$  and  $\overline{G}^*(DB, \overline{G}^*(DB, G, p, q), BK)$  are empty, then both players cannot force each other to check indefinitely inside  $\overline{G}^*(DB, G, p, q)$  and therefore the positions in  $\overline{G}^*(DB, G, p, q)$  can be safely declared as draws.

**Lemma 13** *Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and a subgraph  $G_1 = (V_1, E_1)$  of  $G$  with no terminal vertices and  $\forall v \in V_1, DB(v) = \mathbf{unknown}$ ,*

$$G_1 \cap G_2 \in G^*(DB, G_1, p, q),$$

where  $G_2 = (V_2, E_2) \in G^*(DB, G, p, q)$  and  $p, q$  are two pieces of different sides.

**Proof** By Lemma 3,  $G_2$  is an induced subgraph of  $G$ . Therefore,  $G_1 \cap G_2$  is an induced subgraph of  $G_1$ . Since  $G_2 \in G^*(DB, G, p, q)$ ,  $G_1 \cap G_2$  satisfies the first five conditions in Definition 5 for being a move pattern in  $G^*(DB, G_1, p, q)$ . Now we investigate the last condition. Denote  $G_1 \cap G_2 = (\overline{V}, \overline{E})$ . Given  $u \in \overline{V} \subseteq V_1$ , there exists  $(u, v) \in E_1$  with  $DB(v) = \mathbf{unknown}$ , since  $G_1$  has no terminal vertices. By conditions (1), (4) and (5) in Definition 5,  $(u, v) \in E_2$ . Therefore,  $(u, v) \in \overline{E}$ , which implies that the graph  $G_1 \cap G_2$  does not have terminal vertices. The last condition in Definition 5 also holds. As a result,  $G_1 \cap G_2 \in G^*(DB, G_1, p, q)$ .  $\square$

**Theorem 5** *Given a win-draw-loss-unknown database  $DB$  for a game graph  $G$  and two pieces  $p, q$  of different sides,*

$$\overline{G}^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, r) = \emptyset$$



for any piece  $r$ .

**Proof** Without loss of generality, we assume that the two pieces  $q$  and  $r$  are of the same player.  $\overline{G}_\dagger^*(DB, G, r)$  and  $\overline{G}^*(DB, G, p, q)$  satisfy the requirements for being  $G_1$  and  $G_2$  in Lemma 13. Therefore, we obtain  $\overline{G}^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, r) \in G^*(DB, \overline{G}_\dagger^*(DB, G, r), p, q) \subseteq G^*(DB, \overline{G}_\dagger^*(DB, G, r), p)$ . By Definition 7,  $G^*(DB, \overline{G}_\dagger^*(DB, G, r), p) = \emptyset$ . Therefore,  $\overline{G}^*(DB, G, p, q) \cap \overline{G}_\dagger^*(DB, G, r) = \emptyset$ .  $\square$

Given a win-draw-loss-unknown database  $DB$  for graph  $G$ , we use  $G_0$  to denote the union of the maximum move patterns of totally non-mutual chasing indefinitely  $\overline{G}^*(DB, G, p, q)$  satisfying the conditions  $\overline{G}^*(DB, \overline{G}^*(DB, G, p, q), RK) = \emptyset$  and  $\overline{G}^*(DB, \overline{G}^*(DB, G, p, q), BK) = \emptyset$  for all pairs of pieces  $p, q$  of different sides other than the Kings. By Theorems 4 and 5,  $G_1, G_2$  in Algorithm 6 and  $G_0$  are all disjoint. Therefore, the positions in them can be correctly marked with their win-draw-loss information without conflicts.

Draw positions can be propagated. In Western chess, it has zero practical value, since the draw positions are eventually all marked in the final phase. In Chinese chess, it may improve the efficiency for computing the maximum move patterns, since the number of candidates initialized for computing the maximum move patterns in the later iterations may be reduced.

Given a database  $DB$  for graph  $G = (V, E)$ , a set of seed draw positions  $D \subseteq V$  is called *valid* if all vertices in  $D$  are unknown or draw positions in  $DB$ , and after marking all these vertices as draws in  $DB$ ,  $DB$  remains a sound win-draw-loss database (i.e., satisfying all the conditions in Definition 3). Given a set of valid seed draw positions, the draw information is iteratively propagated back to the predecessors until no propagation is possible. The pseudo-code is given in Algorithm 7.

---

#### Algorithm 7 Set Draws With Propagation

---

**Require:**  $D$  is a valid set of seed draw positions.

**Ensure:**  $DB$  remains a sound win-draw-loss-unknown database when job done.

```

procedure SETDRAWS( $DB$  as a database,  $G = (V, E)$  as a graph,  $D \subseteq V$  as a set of seed draws)
  for all  $u \in D$  do ▷ Initialization Phase
     $DB \leftarrow \mathbf{draw}$ 
  end for
  while  $D \neq \emptyset$  do ▷ Propagation Phase
    Pop any  $u \in D$  and set  $D \leftarrow D - \{u\}$ .
    for all  $(v, u) \in E$  with  $DB(v) = \mathbf{unknown}$  do
      if  $\forall (v, w) \in E, (DB(w) = \mathbf{draw}) \vee (DB(w) = \mathbf{win})$  then
         $D \leftarrow D \cup \{v\}$ 
         $DB(v) \leftarrow \mathbf{draw}$ 
      end if
    end for
     $D \leftarrow D - \{u\}$ 
  end while
end procedure

```

---

Four remarks for the propagation of draws are given below. First, given a database  $DB$  for graph  $G$ , any move pattern of mutual checking/chasing indefinitely in  $G^*(DB, G, p, q)$  for two pieces  $p, q$  of different sides is a valid set of seed draw positions. Second, during the propagation, the  $DB$  keeps being a sound win-draw-loss-unknown database. Third, the property that  $DB$  is semi-fully-propagated is preserved, because updating the draw positions does not affect being semi-fully-propagated. Fourth, the propagation of draws and the propagation of wins and losses are independent. An unknown position is marked as a win if it has a child as a loss. An unknown position is marked as a loss if its children are all wins. An unknown position is marked as a draw if it has a draw child and its children are all draws or wins.

In Algorithm 6, there are three types of valid sets of seed draw positions: maximum move patterns of mutual checking indefinitely, maximum move patterns of mutual chasing indefinitely, and the terminal draw positions. In Western chess, the terminal draw positions are stalemate positions. In Chinese chess, there are no terminal draw positions, unless the game graph is split. When the game graph is split, the terminal draw positions are those propagated from the supporting databases.

**Lemma 14** Given a database  $DB$  for graph  $G = (V, E)$  and a valid set of seed draw positions  $D$ , we denote  $DB_1$  as the database of  $DB$  after marking all vertices in  $D$  as draws, and  $DB_2$  as the database after propagating draws by Algorithm 7, then

$$G^*(DB_1, G, p) = G^*(DB_2, G, p) \text{ and } G^*(DB_1, G, p, q) = G^*(DB_2, G, p, q)$$

for any pieces  $p$  and  $q$  of different sides. In other words, the propagation of draws do not affect the move patterns of (mutual) checking/chasing indefinitely.

**Proof** Both players in the propagated draw positions can force each other to some positions in  $D$  or some draw position in  $DB$ . Given any position  $u$  with  $DB_1(u) = \mathbf{unknown}$  and  $DB_2(u) = \mathbf{draw}$ , no matter the next mover in  $u$  is the attacker or defender,  $u$  cannot be a position in  $\overline{G^*}(DB_1, G, p, q)$ , because both players can force each other to lead the game to some position in  $D$  or some draw position in  $DB$ , so condition (5) in Definition 4 cannot hold. Therefore,  $G^*(DB_1, G, p) = G^*(DB_2, G, p)$ . A similar discussion gets  $G^*(DB_1, G, p, q) = G^*(DB_2, G, p, q)$ .  $\square$

The consequent products of Lemma 14 are as follows. First, because  $G^*(DB_1, G, p, q) = \overline{G^*}(DB_2, G, p, q)$ ,  $\overline{G^*}(DB_1, G, p, q) = \overline{G^*}(DB_2, G, p, q)$ . Second,  $G^*(DB_1, G, p) = G^*(DB_2, G, p)$ , so  $\overline{G^*}(DB_1, G, p) = \overline{G^*}(DB_2, G, p)$ . Third,  $\overline{G^*}_\dagger(DB_1, G, p, q) = \overline{G^*}_\dagger(DB_2, G, p, q)$ , because  $\overline{G^*}(DB_1, G, p) = \overline{G^*}(DB_2, G, p)$  and therefore reductions in Algorithm 4 are the same. Fourth,  $\overline{G^*}_\dagger(DB_1, G, p) = \overline{G^*}_\dagger(DB_2, G, p)$  is obtained similarly. As a result, the propagations of draws in Algorithm 6 do not change the resulting built database. However, it can reduce the cost for computing the maximum move patterns because there may be fewer candidates in the initialization phase. Whether it is worthwhile to trade the cost of computing and propagating draws for the saving of computing maximum move patterns of checking/chasing indefinitely may depend on the endgame databases and the implementations.

In the selected 50 endgame databases in (Fang *et al.*, 2004; Fang, 2004), the maximum move patterns of mutual checking/chasing indefinitely computed in Algorithm 6 are empty. Therefore, the only seed draws are the terminal draws propagated from the supporting databases when the graph is split. At this stage, the dealing with draws from the maximum move patterns of mutual checking/chasing indefinitely is of theoretical interest only.

**Theorem 6** The database constructed by Algorithm 6 is a sound win-draw-loss database (i.e., satisfying all conditions in Definition 3).

**Proof** To verify the soundness of the win-draw-loss database constructed by Algorithm 6, we inspect the three types of positions: non-draw seeds, propagated non-draw positions, and the draws.

1. There are three types of non-draw seeds: terminal positions, loss positions in maximum move patterns of non-mutual checking indefinitely, and loss positions in maximum move patterns of non-mutual chasing indefinitely. For each win position of non-mutual checking (or chasing) indefinitely, it must have a child as a loss position in the same pattern. During propagation of checking (or chasing) indefinitely, they are all correctly marked as wins. Therefore, they satisfy conditions (1) and (2) in Definition 3.
2. Inspecting the propagation phase in Algorithm 6, each propagated win position must have at least one loss child. Each propagated loss position must have all children as win positions. Both conditions (1) and (2) in Definition 3 are satisfied.
3. When no propagation is possible, each unknown non-terminal position must have at least one unknown or draw child and no loss children. Therefore, the draws marked in the final phase satisfy the condition (3) in Definition 3. The draws in the maximum move patterns of mutual checking (or chasing) indefinitely marked prior to the final phase still satisfy condition (3) in Definition 3, and so do the propagated draws.  $\square$

## 5. INFALLIBLE COMPLETE CHINESE CHESS ENDGAME DATABASES

The complete Chinese-chess endgame databases constructed by Algorithm 6 contains the win-draw-loss information in accord with the Asian rule set. However, with only this win-draw-loss information available,

one player may rove in the win states but never win the game by checkmate, stalemate, or forcing his opponent to check or chase indefinitely. How infallibility is achieved with the complete endgame databases is described as follows. Subsection 5.1 gives an algorithm to build endgame databases in the distance-to-mate/check/chase metric. Subsection 5.2 describes the infallibility playing strategy. Subsection 5.3 adapts the algorithm for endgame databases in the distance-to-conversion/check/chase metric. Subsection 5.4 describes how to verify the endgame databases to ensure that there are no hardware and software errors.

## 5.1 Distance-to-mate/check/chase Endgame Databases

In the endgame databases of Chinese chess in the *distance-to-mate* metric, each position value is represented as the distance to the checkmate or stalemate positions measured in plies. We note that checkmate and stalemate positions are the loss positions with distance 0. All win/loss positions have odd/even distance values, respectively. With the rules of checking and chasing indefinitely, the shortest win can hardly be defined. Hence we concentrate on the infallibility.

In (Fang *et al.*, 2002; Fang *et al.*, 2004), the concept of *distance-to-check* was introduced, which is defined as the distance measured in plies to the loss positions of checking indefinitely (i.e., the loss positions in  $\overline{G}^*(DB, G, RK)$  and  $\overline{G}^*(DB, G, BK)$ ). In this paper, it is revised to be the distance to the loss positions of non-mutual checking indefinitely<sup>11</sup> (i.e., the loss positions in  $\overline{G}_\dagger^*(DB, G, RK, BK)$  and  $\overline{G}_\dagger^*(DB, G, BK, RK)$ ). Similarly, the term *distance-to-chase* is defined as the distance to the loss positions in the maximum move pattern of totally non-mutual chasing indefinitely measured in plies<sup>12</sup> (i.e., the loss positions in  $\overline{G}_\dagger^*(DB, G, p)$ , where  $p$  is not a King).

The propagation phase for non-mutual checking (or chasing) indefinitely positions is similar to that in the distance-to-mate metric. Note that all odd/even distance values in distance-to-check and distance-to-chase metrics also represent the win/loss positions, respectively. In the resulting database, some positions are in distance-to-mate metric, some others are in distance-to-check metric, and some others are in distance-to-chase metric. Therefore, the database is in *distance-to-mate/check/chase* metric. As discussed in Subsection 4.1, non-mutual checking indefinitely and non-mutual chasing indefinitely have their levels. This suggests the definition of a position value as follows.

**Definition 8** For the complete endgame databases of Chinese chess in the *distance-to-mate/check/chase* metric, a position value consists of two integers, denoted by  $(r, d)$ , where  $r$  is the level and  $d$  is the distance measured in plies. When  $r = 0$ ,  $d$  stands for distance-to-mate. When  $r > 0$ ,  $d$  stands for distance-to-check or distance-to-chase. Draw and unknown positions have their specific position values other than any possible position values in distance-to-mate, distance-to-check and distance-to-chase in practice. Moreover, function  $f$  is to map position values to  $\{\mathbf{win}, \mathbf{draw}, \mathbf{loss}, \mathbf{unknown}\}$  defined by:

$$f((r, d)) = \begin{cases} \mathbf{win} & \text{if } r \geq 0 \text{ and } d \text{ is odd.} \\ \mathbf{loss} & \text{if } r \geq 0 \text{ and } d \text{ is even.} \\ \mathbf{draw} & \text{if } (r, d) \text{ represents a draw.} \\ \mathbf{unknown} & \text{if } (r, d) \text{ represents an unknown.} \end{cases}$$

Algorithm 8, modified from Algorithm 6, is to build endgame databases in distance-to-mate/check/chase metric. The win-draw-loss information is identical to that of the databases built by Algorithm 6 as stated in the following theorem.

**Theorem 7** Let  $DB$  be a database in distance-to-mate/check/chase metric constructed by Algorithm 8.  $f(DB)$  is identical to the database constructed by Algorithm 6.

**Proof** Comparing Algorithm 8 with Algorithm 6, it is not hard to see the identical property after a little thought.  $\square$

<sup>11</sup>In practical cases, they are usually identical. For example, in the selected 50 endgame databases in (Fang *et al.*, 2004; Fang, 2004), the set of the move patterns of checking indefinitely is the same as that of non-mutual checking indefinitely.

<sup>12</sup>Another definition of distance-to-check (distance-to-chase) is the distance to the win positions of non-mutual chasing (totally non-mutual checking) indefinitely. It also results in the infallible endgame databases by a similar algorithm.

**Algorithm 8** Building  $DB$  in distance-to-mate/check/chase metric for  $G = (V, E)$ 


---

```

for all  $v \in V$  do ▷ Initialization Phase
  Set  $DB(v)$  so that  $f(DB(v)) = \mathbf{unknown}$ .
end for
 $W \leftarrow \{\text{terminal win positions}\}$ 
 $L \leftarrow \{\text{terminal loss positions}\}$ 
 $r \leftarrow 0$ 
repeat
   $d \leftarrow 0$ 
  repeat ▷ Propagation Phase
    if  $d$  is even then ▷ propagating loss positions
      for all  $u \in L$  do
         $DB(u) \leftarrow (r, d)$ 
        for all  $(v, u) \in E$  with  $f(DB(v)) = \mathbf{unknown}$  do
           $W \leftarrow W \cup \{v\}$ 
        end for
      end for
       $L \leftarrow \emptyset$ 
    else [ $d$  is odd] ▷ propagating win positions
      for all  $v \in W$  do
         $DB(v) \leftarrow (r, d)$ 
        for all  $(u, v) \in E$  with  $f(DB(u)) = \mathbf{unknown}$  do
          If  $\forall (u, w) \in E, f(DB(w)) = \mathbf{loss}$ , then  $L \leftarrow L \cup \{u\}$ .
        end for
      end for
       $W \leftarrow \emptyset$ 
    end if
     $d \leftarrow d + 1$ 
  until  $W = \emptyset$  and  $L = \emptyset$ 
   $r \leftarrow r + 1$  ▷  $DB$  is semi-fully-propagated here.
  {In the first iteration, optionally marking terminal seed draws with propagation here.}
  {Optionally marking positions of mutual checking indefinitely with propagation here.}
  if  $\overline{G}_\dagger^*(f(DB), G, RK, BK) \cup \overline{G}_\dagger^*(f(DB), G, BK, RK) \neq \emptyset$  then ▷ non-mutual checking indefinitely
     $L \leftarrow \{\text{loss positions in } \overline{G}_\dagger^*(f(DB), G, RK, BK) \cup \overline{G}_\dagger^*(f(DB), G, BK, RK)\}$ 
  else
    {Optionally marking positions of mutual chasing indefinitely with propagation here.}
     $G_1 \leftarrow \text{union of all } \overline{G}_\dagger^*(f(DB), G, p) \text{ with } p \text{ as a Red piece other than the King}$ 
     $G_2 \leftarrow \text{union of all } \overline{G}_\dagger^*(f(DB), G, q) \text{ with } q \text{ as a Black piece other than the King}$ 
    if  $G_1 \cup G_2 \neq \emptyset$  then ▷ non-mutual chasing indefinitely
       $L \leftarrow \{\text{loss positions in } G_1 \cup G_2\}$ 
    else
      Break the repeat loop. ▷ (*)
    end if
  end if
  until the repeat loop breaks in (*).
  for all  $v \in V$  with  $f(DB(v)) = \mathbf{unknown}$  do ▷ Final Phase
    Set  $DB(v)$  so that  $f(DB(v)) = \mathbf{draw}$ .
  end for

```

---

**5.2 Infallible Playing Scheme**

The infallible playing scheme and the relative theorems and lemmas are analogous to those in (Fang *et al.*, 2004) with the databases in distance-to-mate/check metric. The *order* of win-draw-loss information is defined by **win** > **draw** > **loss**. For win/loss positions in the distance-to-mate/check/chase metric, the

lower the level, the better/worse the position, respectively. For win/loss positions at the same level, the shorter the distance, the better/worse the position, respectively. The precise definition is as follows.

**Definition 9** Given two position values  $(r_1, d_1)$  and  $(r_2, d_2)$  in distance-to-mate/check/chase metric, their order is defined by the following rules.

- If  $(r_1 = r_2) \wedge (d_1 = d_2)$ , then  $(r_1, d_1) = (r_2, d_2)$ .
- If  $f((r_1, d_1)) > f((r_2, d_2))$ , then  $(r_1, d_1) > (r_2, d_2)$ .
- If  $(f((r_1, d_1)) = f((r_2, d_2)) = \mathbf{win}) \wedge (r_1 < r_2)$ , then  $(r_1, d_1) > (r_2, d_2)$ .
- If  $(f((r_1, d_1)) = f((r_2, d_2)) = \mathbf{win}) \wedge (r_1 = r_2) \wedge (d_1 < d_2)$ , then  $(r_1, d_1) > (r_2, d_2)$ .
- If  $(f((r_1, d_1)) = f((r_2, d_2)) = \mathbf{loss}) \wedge (r_1 > r_2)$ , then  $(r_1, d_1) > (r_2, d_2)$ .
- If  $(f((r_1, d_1)) = f((r_2, d_2)) = \mathbf{loss}) \wedge (r_1 = r_2) \wedge (d_1 > d_2)$ , then  $(r_1, d_1) > (r_2, d_2)$ .
- Otherwise,  $(r_1, d_1) < (r_2, d_2)$ .

This order is well-defined since  $((r_1, d_1) > (r_2, d_2)) \wedge ((r_2, d_2) > (r_3, d_3)) \implies ((r_1, d_1) > (r_3, d_3))$ . In addition, we call  $(r_1, d_1) \geq (r_2, d_2)$  if  $((r_1, d_1) > (r_2, d_2)) \vee ((r_1, d_1) = (r_2, d_2))$ , and  $(r_1, d_1) \leq (r_2, d_2)$  if  $((r_1, d_1) < (r_2, d_2)) \vee ((r_1, d_1) = (r_2, d_2))$ .

**Theorem 8** In the database  $DB$  in distance-to-mate/check/chase metric constructed by Algorithm 8 of a fully-extended subgraph  $G = (V, E)$  of Chinese chess,

- $\forall$  propagated  $u \in V$  with  $DB(u)$  denoted by  $(r, d)$  with  $d > 0$ ,  $\exists(u, v) \in E$  such that  $DB(v) = (r, d - 1)$ . In addition,  $\forall(u, w) \in E$ ,  $DB(w) \geq (r, d - 1)$ .
- $\forall u \in V$  satisfying  $DB(u) = (r, 0)$  with  $r > 0$ ,  $\exists(u, v) \in E$  such that  $DB(v) = (r, 1)$ . In addition,  $\forall(u, w) \in E$ ,  $DB(w) \geq (r, 1)$ .

**Proof** The proof is much the same as that of (Fang *et al.*, 2004, Theorem 3). In Algorithm 8, a propagated  $u \in V$  is either win or loss. If  $u$  is a win position,  $DB(u)$  is set whenever a loss child, denoted by  $v$ , is found and added into  $L$  in the previous iteration, so that  $DB(v) = (r, d - 1)$ . Since other loss children are determined in the same or later iteration, they are either at a higher level, or at the same level with a longer distance. By Definition 9, they are  $(r, d - 1)$  or better. If  $u$  is a loss position,  $DB(u)$  is set whenever all children are known as win positions. The latest known child, denoted by  $v$ , is added into  $W$  in the previous iteration, so that  $DB(v) = (r, d - 1)$ . All other children are loss and determined in the same or later iteration, they are either at a lower level, or at the same level with a shorter distance. By Definition 9, they are equal to or better than  $(r, d - 1)$ .

Each position  $u$  satisfying  $DB(u) = (r, 0)$  for some  $r > 0$  is a loss position in some maximum pattern of checking (or chasing) indefinitely, so that one of its children is a win position in this pattern. Besides, all the win positions in the same pattern are propagated as  $(r, 1)$ , since they must have a loss child with position value  $(r, 0)$ . All other children of  $u$  are win positions at the lower level determined in the previous iterations, which is greater than  $(r, 1)$  by Definition 9. As a result,  $\forall(u, w) \in E$ ,  $DB(w) \geq (r, 1)$ .  $\square$

The infallible playing scheme with the complete endgame databases in distance-to-mate/check/chase metric for a winning player is similar to that in (Fang *et al.*, 2004). Starting from a win position  $v$  with  $DB(v)$  denoted by  $(r, d)$  where  $d$  is odd, always move to some child  $u$  satisfying  $DB(u) = (r, d - 1)$ . By Theorem 8, if  $d > 2$ , all the moves of the opponent lead to win positions  $(r, d - 2)$  or better (i.e., either the level is lower than  $r$  or it is the same level but the distance is no longer than  $d - 2$ ). As a result, the winning player can always force his opponent to be in a position  $(r', 0)$  for some  $r'$  satisfying  $0 \leq r' \leq r$ . If  $r' = 0$ , the winning player checkmates or stalemates his opponent. For  $r' > 0$ , the losing player is forced to check (or chase) the winning side indefinitely, if he always moves to the position  $(r', 1)$ . Otherwise, the losing player is forced to move to some position at a level lower than  $r'$  and the winning player keeps in a win position. Note that if the level  $r' > 0$  is of non-mutual chasing indefinitely, problems may occur while game continues in the positions of  $(r', 0)$  and  $(r', 1)$ . Without loss of generality, we assume that Red is the attacker. The defender may chase a piece  $p_1$  every other move and chase another piece  $p_2$  every the other

move, and therefore the game results in a draw (Association, 1999, page 103, rule 32). This is because in Algorithm 8, at level  $r'$ ,  $G_1 \subseteq \overline{G}_\dagger^*(DB, G, p_1) \cup \overline{G}_\dagger^*(DB, G, p_2)$ , but  $\overline{G}_\dagger^*(DB, G, p_1) \cap \overline{G}_\dagger^*(DB, G, p_2) \neq \emptyset$ . One way to avoid this problem is by adding an attribute to each position of  $(r', 1)$  indicating which maximum move pattern(s) of totally non-mutual chasing indefinitely it belongs to. Since both level and distance are bounded below, the winning side can always win the game by checkmate, stalemate, or forcing his opponent to check or chase indefinitely.

### 5.3 Distance-to-Conversion/Check/Chase Metric

The game graph of Western chess or Chinese chess is usually split according to the numbers of different pieces on the board into multiple endgame databases. The *distance-to-conversion* metric was introduced to shorten the maximal distance. Conceptually, it is the distance to the supporting databases measured in plies. The resulting endgame databases are also infallible.

In (Fang *et al.*, 2004), the distance-to-conversion strategy is also incorporated into the algorithm to construct the endgame databases in the *distance-to-conversion/check* metric. For databases with complete information, endgame databases in the *distance-to-conversion/check/chase* metric can be built as follows. In addition to checkmate and stalemate positions, the loss terminal positions are re-defined to include those with all children being loss positions in the supporting databases. The win terminal positions are re-defined to be those with a loss child in the supporting databases. This scheme results in *distance-to-conversion/check/chase* metric. It can also be applied to Algorithm 6 for complete win-draw-loss endgame databases, and all the theorems and lemmas in Sections 3 and 4 remain sound.

For databases in distance-to-conversion/check/chase metric, the following definitions (see 1, below) and algorithm (see 2, below) are the same, and the theorems (see 3, below) remain sound, except that each "mate" is replaced by "conversion", and the terminal win and loss positions are re-defined as above.

1. The position values and their order in Definitions 8 and 9, respectively.
2. Algorithm 8 to build an endgame database.
3. Theorem 7 for the relationship with the win-draw-loss database built by Algorithm 6, and Theorem 8 for infallible playing strategy.

The playing strategy with the endgame databases in distance-to-conversion/check/chase metric remains infallible. Through a similar discussion in Subsection 5.2, starting from a win position  $v$  with  $DB(v)$  denoted by  $(r, d)$  in distance-to-conversion/check metric, the winning player can always either force the game to fall into some supporting database and keep himself as the winning side, or win the game by checkmate, stalemate, or forcing his opponent to check or chase indefinitely. Since falling into a supporting database cannot be repeated endlessly, the winning side is guaranteed to win the game infallibly.

### 5.4 Verification Algorithm

Constructing endgame databases is a time-consuming computational task. Verification is required to ensure the correctness without hardware or software errors. Given a database  $DB$  for a fully-extended subgraph of the Chinese chess game graph in the distance-to-mate/check/chase or the distance-to-conversion/check/chase metric, the verification consists of confirming three types of positions: the non-draw seeds, the propagated non-draw positions, and the draws.

1. The non-draw seeds are verified by the same methods used to determine them. Seeds at the zero-th level are verified by confirming that they are the right terminal positions. Seeds for non-mutual checking (or chasing) indefinitely are verified by confirming that they are the loss positions in the maximum patterns of non-mutual checking (or chasing) indefinitely at their levels.
2. Propagated win and loss positions in  $DB$  are verified by Theorem 8.
3. Draw positions are verified with  $f(DB)$  by Definition 3, where function  $f$  is defined in Definition 8.

## 6. CONCLUDING REMARKS

Retrograde analysis has been widely used and successfully applied to construct endgame databases of Western chess. In Chinese chess, its application is limited because of the special rules. After tackling the problems caused by the special rules, an algorithm is successfully developed for building the complete Chinese-chess endgame databases in accord with the Asian rule set. The concluding remarks are as follows.

1. To verify the completeness of the endgame databases, the rule-tolerant approach works well in practice for both the Asian rule set and the Chinese rule set (Fang, 2004). To build complete endgame databases, we need rigorous algorithms. To comply with the path-dependent rule (Association, 1999, page 103, rule 32) in the Asian rule set, the function  $chase((u, v), p)$  is defined to indicate whether  $(u, v)$  is a move to chase  $p$  or not. However, this rule does not exist in the Chinese rule set. Besides, the Chinese rule set has different path-dependent rules. Therefore, this approach does not work for the Chinese rule set.
2. Given a win-draw-loss-unknown database  $DB$  for graph  $G$  and pieces  $p, q$  of different sides, the maximum move patterns  $\overline{G^*}(DB, G, p)$ ,  $\overline{G^*}(DB, G, p, q)$ ,  $\overline{G^*}_\dagger(DB, G, p, q)$  and  $\overline{G^*}_\dagger(DB, G, p)$  are all induced subgraphs of  $G$ . Using this fact, the algorithms to compute these maximum move patterns consist of an initialization phase and a pruning phase. In the initialization phase, a supergraph is computed. In the pruning phase, unqualified vertices are pruned, until the graph induced by the remaining vertices is the maximum move pattern.
3. The children counting strategy can be applied to not only Algorithms 6 and 8, but also Algorithms 1 and 3, which are invoked by Algorithms 4 and 5. With the children counting strategy applied, each edge is visited at most once during the propagation phases or the pruning phase.
4. In Algorithm 6 for building the complete win-draw-loss endgame databases, the database  $DB$  keeps being a sound win-draw-loss-unknown endgame database. The  $DB$  is semi-fully-propagated after each propagation phase. If no draw positions are marked prior the final phase, then being semi-fully-propagated is equivalent to being fully-propagated.
5. By Theorem 4, the maximum move patterns of non-mutual checking indefinitely and mutual checking indefinitely are disjoint, and maximum move patterns of totally non-mutual chasing indefinitely and mutual chasing indefinitely are disjoint in Algorithms 6 and 8. Therefore, the positions in them can be marked with their win-draw-loss information without conflicts.
6. In Algorithms 6 and 8, the draw positions in the maximum move patterns of mutual checking/chasing indefinitely do not need to be taken care of, since they are eventually marked as draws in the final phase. The draws can be iteratively propagated by Algorithm 7. Marking these draws may improve the efficiency for computing the maximum move patterns, since there may be fewer candidates initialized.
7. For infallible playing strategy, the databases in the distance-to-mate/check/chase metric are introduced. It is similar to the strategy for the databases in the distance-to-mate/check metric in (Fang *et al.*, 2004). While the game graph is split, the databases are in the distance-to-conversion/check/chase metric. The theorems and lemmas remain sound with the terminal vertices re-defined. The verification algorithm to ensure the correctness is similar to that in (Fang *et al.*, 2004) for databases in the distance-to-mate/check metric or the distance-to-conversion/check metric.

## ACKNOWLEDGEMENT

The author is very grateful to Dianne P. O’Leary for editing this article.

## 7. REFERENCES

Association, C. X. (1999). *The Playing Rules of Chinese Chess*. Shanghai Lexicon Publishing Company. In Chinese.

Fang, H.-r. (2004). Rule-tolerant Verification Algorithms for Completeness of Chinese Chess Endgame Databases. Accepted by CG'04 conference. To appear.

Fang, H.-r., Hsu, T.-s., and Hsu, S.-c. (2000). Construction of Chinese Chess Endgame Databases by Retrograde Analysis. *Lecture Notes in Computer Science 2063: Proceedings of the 2nd International Conference on Computers and Games* (eds. T. Marsland and I. Frank), pp. 96–114. Springer-Verlag, New York, NY.

Fang, H.-r., Hsu, T.-s., and Hsu, S.-c. (2002). Indefinite Sequence of Moves in Chinese Chess Endgames. *Lecture Notes in Computer Science 2063: Proceedings of the 3rd International Conference on Computers and Games* (eds. J. Schaeffer, M. Müller, and Y. Björnsson), pp. 264–279. Springer-Verlag, New York, NY.

Fang, H.-r., Hsu, T.-s., and Hsu, S.-c. (2004). Checking Indefinitely in Chinese-Chess Endgames. *ICCA Journal*, Vol. 27, No. 1, pp. 19–37.

Herik, H. J. van den, Uiterwijk, J. W. H. M., and Rijswijk, J. van (2002). Games Solved: Now and in the Future. *Artificial Intelligence*, Vol. 134, pp. 277–311. ISSN 0004–3702.

Romein, J. and Bal, H. (2003). Solving the Game of Awari using Parallel Retrograde Analysis. *IEEE Computer*, Vol. 36, No. 10, pp. 26–33.

Thompson, K. (1996). 6-Piece Endgames. *ICCA Journal*, Vol. 19, No. 4, pp. 215–226.

Wu, R. and Beal, D. (2001). Fast, Memory-Efficient Retrograde Algorithms. *ICCA Journal*, Vol. 24, No. 3, pp. 147–159.