mathematics

MDPI

Article

On Primitive Recursive Characteristics of Chess

Vladimir Kulyukin

Department of Computer Science, Utah State University, Logan, UT 84322, USA; vladimir.kulyukin@usu.edu

Abstract: Several characteristics of chess are investigated with methods of computability and number theories. It is shown that for an unfinished game it is primitive recursively decidable whether the game is winnable, drawable, or absolutely losable within a specified number of future moves for the player whose turn it is to play on the last board of the game. It is also shown that there exist primitive recursive procedures to compute optimal continuations of unfinished games within specified numbers of future moves and that the set of chess games is recursive.

Keywords: computability theory; theory of recursive functions; number theory; game theory; chess

Citation: Kulyukin, V. On Primitive Recursive Characteristics of Chess. *Mathematics* **2022**, *10*, 1016. https://doi.org/10.3390/math10071016

Academic Editor: David Barilla

Received: 12 February 2022
Accepted: 15 March 2022
Published: 22 March 2022

1. Introduction

A deterministic two-player board game (D2PBG) is played by two players on a finite board. The board is finite in the sense that it contains finitely many positions where each player can make a move. The players take turns making moves on the board with exactly one move per player per turn. The order of turns is pre-determined and cannot be changed at any point during the game. At each turn, a move is chosen by a player out of finitely many possible moves and uniquely determines the next board.

A D2PBG is a sequence of boards with a unique starting board. The last board (or the end board) of any such sequence is reached from the starting board after finitely many valid (i.e., allowed by the rules of the game) moves, and can be classified as a win for either player, a draw, or an unfinished board in the sense that the game can continue from its end board for a positive number of moves into the future. Examples of D2PBGs are Tic Tac Toe [1] and numerous variants thereof (e.g., Qubic [2]), chess, and checkers.

We follow Kleene's classification of computational methods into *decision procedures* and *calculation procedures* (Chapter 6 in [3]). A decision procedure is a mechanical method expressed in a symbolic formalism (e.g., λ-calculus [4] or the language L [5]) that returns, in finitely many steps, a positive (e.g., 1) or negative (e.g., 0) answer to a class of problems. We refer to such classes of problems as *characteristics*. Thus, the characteristic of *relative primality* of two positive integers in number theory is decidable by a decision procedure that uses Euclid's algorithm to find their greatest common divisor, checks whether it is equal to 1 and returns 1 (i.e., true) if it is, and 0 (i.e., false) if it is not. The characteristic of *winnability* in the game of Tic Tac Toe is decidable by a decision procedure that for the end board b of any unfinished Tic Tac Toe game returns 1 if it is winnable by either player (i.e., `X` or `O`) whose turn it is to play on b within M moves where M is a positive integer [6] and returns 0 if it is not winnable. As shown in this investigation, the characteristic of *stalemate* for a chess game is decidable by a decision procedure that determines if the end board of the game is a stalemate for the player whose turn it is to play on it. We refer to a feature of a D2PBG as a *decision characteristic* if there exists a decision procedure to decide it.

A calculation procedure requires an effective (i.e., mechanical and deterministic) construction, in finitely many steps, of a mathematical object (e.g., a sequence of instructions in λ-calculus, a plot, a matrix, etc.) that satisfies specific properties. Euclid's algorithm is a calculation procedure for the characteristic of relatively primality, because it constructs in finitely many steps the required object – the greatest common divisor of two numbers. Similarly, a calculation procedure for the game of Tic Tac Toe takes any unfinished game

and, if the latter is winnable within M moves by the player whose turn it is to play on its last board, constructs a sequence of at most M moves that result in a win board for the player. We call a feature of a D2PBG a *calculation characteristic* if there exists a calculation procedure to produce required mathematical objects (e.g., boards or games) with specific properties. The existence of a calculation procedure for a characteristic implies the existence of a decision procedure for it that returns 1 if a required object is constructed and 0 otherwise.

We may require that a decision or calculation procedure itself satisfy certain properties. In general, those properties can be either *structural* (e.g., a procedure must be primitive recursive, computable, or partially computable) or *algorithmic* (e.g., the number of steps the procedure takes to return a decision or construct an object must be the function of the size of the input n such as $O(n^3)$ and must be known ahead of the calculation).

In this article, we focus on the structural (and, more specifically, primitive recursive) characteristics of chess. In that, we follow a fundamental tenet of classical computability theory, which, as formulated by Rogers [4], states that

> ‘‘[w]e thus require that a computation terminate after some finite number of steps; we do not insist on an a priori ability to estimate this number.’’

We thus study the arithmetization of chess (and, by implication, of other D2PBGs) to analyze the game's characteristics with methods of number theory and classical computability. If a decision characteristic of chess can be computed with a primitive recursive predicate, we call the characteristic *primitive recursively decidable* or *pr-decidable*; if a calculation characteristic of chess can be computed with a primitive recursive function, we refer to it as *primitive recursively computable* or *pr-computable*.

Unfortunately, the theory of recursive functions appears to lack a uniform, commonly accepted formalism. The treatment of primitive recursive functions in this article is based on the formalism by Davis et al. in [5], which, in turn, is based on Kleene's formalism in (Chapter 9 in [3]). Alternative treatments are given in [7], where primitive recursive functions are formalized as loop programs consisting of assignment and iteration statements similar to DO statements in FORTRAN, or in [4], where primitive recursive functions are described with λ-calculus.

The words *number* or *numbers* in this article refer to a natural number or natural numbers, respectively, where the set of natural number is defined as $\mathbb{N} = \{0, 1, 2, \ldots\}$. All small letters (e.g., x, y, z, a, b, c, i, j, k, t, l, m, n, u, z, ζ, etc.) with appropriate numeric or symbolic subscripts (when the latter are required by the context) are variables that denote numbers. We sometimes use capital letters (e.g., M, Z) with suitable subscripts or no subscripts to refer to numbers that we believe are significant for the current exposition.

In this article, all functions map $\mathbb{N}^j$ to $\mathbb{N}^k$ (i.e., $\mathbb{N}^j \mapsto \mathbb{N}^k$), where j and k are positive numbers. We use the symbol $\equiv$ to define predicates (e.g., $P(x_1, x_2) \equiv x_1 < x_2$). We use the capital letter P with prime superscripts (e.g., P') or numeric subscripts to define predicates and use the capital letter F with numeric subscripts to define functions. Number 0 when referencing the value of a predicate means that the predicate is false; number 1 when referencing the value of a predicate means that the predicate is true.

When the argument variables of a function or predicate are clear from the context, we use the notation $P(\cdot)$ or $F(\cdot)$ for brevity. For example, $P(\cdot)$ instead of $P(x_1, x_2)$ and $F(\cdot)$ instead of $F(x_1, x_2, x_3)$.

When a variable occurs free in the definition of a predicate or a function it refers to a number referenced in the immediate context. For example, let $z = 10$ and let $P(d) \equiv 2^d | (z + 1)$. In this definition of $P(\cdot)$, z refers to number 10.

In defining some predicates, we occasionally abbreviate Boolean combinations of numerical predicates. For example, $i \leq j \wedge j \leq k$ is abbreviated as $i \leq j \leq k$; $i \geq j \wedge j \geq k$ as $i \geq j \geq k$; $i = j \wedge j = k$ as $i = j = k$, etc.

The remainder of the article is organized as follows. In Section 2, several functions are defined that are shown to be primitive recursive in [5]. In Section 3, we use the primitive

recursive functions in Section 2 to define several primitive recursive functions to manipulate Gödel numbers (G-numbers). We call these functions *G-number operators* and use them in Section 4 to show several characteristics of chess and to be pr-decidable and pr-computable. In Sections 6 and 7, we discuss and summarize our findings.

2. Preliminaries

Let $f(x_1, \ldots, x_k)$ be a function and $g_1, \ldots, g_k$ be functions, where each g_i, $1 \leq i \leq k$, is a function of n arguments. Then $h(x_1, \ldots, x_n)$ is obtained by composition from $f, g_1, \ldots, g_k$ if

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)). \tag{1}$$

Let $k \in \mathbb{N}$ and $\phi(x, y) : \mathbb{N}^2 \mapsto \mathbb{N}$ be total. Let $f(x_1, \ldots, x_n) : \mathbb{N}^n \mapsto \mathbb{N}$ and $g(x_1, \ldots, x_{n+2}) : \mathbb{N}^{n+2} \mapsto \mathbb{N}$ be total. If h is obtained from ϕ by the recurrences in (2) or from f and g by the recurrences in (3), then h is obtained from ϕ or from f and g by *primitive recursion* or, simply, by *recursion*.

$$\begin{array}{rcl} h(0) & = & k, \\ h(t+1) & = & \phi(t, h(t)) \end{array} \tag{2}$$

$$\begin{array}{rcl} h(x_1, \ldots, x_n, 0) & = & f(x_1, \ldots, x_n), \\ h(x_1, \ldots, x_n, t+1) & = & g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n) \end{array} \tag{3}$$

Let

$$\begin{array}{l} s(x) = x + 1; \\ n(x) = 0; \\ u_i^n(x_1, \cdots, x_n) = x_i, \quad 1 \leq i \leq n, \end{array} \tag{4}$$

be the three *initial* functions. A function is *primitive recursive* if it can be obtained from the initial functions by a finite number of applications of composition or recursion as defined in (1)–(3) [5].

A total function $P(x_1, \ldots, x_n)$ is a *predicate* if for any n numbers $y_1, \ldots, y_n$ $P(y_1, \ldots, y_n) = 1$ or $P(y_1, \ldots, y_n) = 0$, where 1 designates *true* and 0 *false*. The expression $(\exists t)_{\leq z}$ is a *bounded existential quantifier* so that if $P(t, x_1, \ldots, x_n)$ is a predicate, $(\exists t)_{\leq z} P(t, x_1, \ldots, x_n) = 1$, if $P(t, x_1, \ldots, x_n) = 1$, for at least one t such that $0 \leq t \leq z$. The expression $(\exists t)_{\leq z} P(t, x_1, \ldots, x_n)$ is called the *bounded existential quantification* of $P(\cdot)$.

The expression $(\forall t)_{\leq z}$ is a *bounded universal quantifier* so that, if $P(t, x_1, \ldots, x_n)$ is a predicate, then $(\forall t)_{\leq z} P(t, x_1, \ldots, x_n) = 1$, if $P(t, x_1, \ldots, x_n) = 1$, for every t such that $0 \leq t \leq z$. The expression $(\forall t)_{\leq z} P(t, x_1, \ldots, x_n)$ is called the *bounded universal quantification* of $P(\cdot)$.

If $P(t, x_1, \ldots, x_n)$ is a predicate and z is a number, then

$$y = \min_{t \leq z} P(t, x_1, \ldots, x_n)$$

is called the *bounded minimalization* of $P(\cdot)$ and defines the smallest number $0 \leq t \leq z$ for which the predicate $P(\cdot)$ is true or 0 if there is no such number. If

$$\min_{t \leq z} P(t, x_1, \ldots, x_n) = 0$$

and $P(0, x_1, \ldots, x_n) = 1$, then 0 returned by the bounded minimalization of $P(\cdot)$ is the smallest number $\leq z$ for which $P(\cdot)$ holds. If $P(0, x_1, \ldots, x_n) = 0$, then 0 returned by the minimalization of $P(\cdot)$ means that there is no number $0 \leq t \leq z$, for which $P(t, x_1, \ldots, x_n)$ holds. If this additional check on 0 must be avoided in a given context, t in the minimalized predicate $P(\cdot)$ must be required to be greater than 0 (e.g., $P(t, x_1, x_2) \equiv t \leq x_1 \cdot x_2 \wedge t > 0$), in which case, 0 returned by the minimalization of $P(\cdot)$ means that there is no $0 \leq t \leq z$ for which $P(\cdot)$ holds. The operator `min` is similar to Kleene's operator μ [3]. If $P(x_1, \ldots, x_n)$ is a n-ary predicate, we occasionally write $P(x_1, \ldots, x_n)$ to abbreviate $P(x_1, \ldots, x_n) = 1$ and write $\neg P(x_1, \ldots, x_n)$ to abbreviate $P(x_1, \ldots, x_n) = 0$.

It is shown in [5] that 1) the predicates $x = y$, $x \neq y$, $x < y$, $x > y$, $x \leq y$, $x \geq y$, and $x|y$ (x divides y or x is a divisor of y) are primitive recursive; 2) any Boolean combination of primitive recursive predicates is primitive recursive; and 3) if a predicate $P(\cdot)$ is primitive recursive, then so are its negation and its bounded minimalization and bounded universal and existential quantifications. We will use the notation $x \nmid y$ to denote $\neg\{x|y\}$ (i.e., x does not divide y or x is not a divisor of y).

Let $x \dot{-} y = x - y$, if $x \geq 0$, and $x \dot{-} y = 0$, if $x < y$. The function $x \dot{-} y$ is shown to be primitive recursive in [5]. Let the pairing of numbers x and y be defined as

$$\langle x, y \rangle = z, \tag{5}$$

where

$$z = 2^x(2y + 1) \dot{-} 1;$$

$$\gamma(d) \equiv \{2^d | (z + 1) \wedge (\forall c)_{\leq z+1}\{2^c \nmid (z + 1) \vee c \leq d\}\};$$

$$x = \min_{d \leq z+1} \gamma(d);$$

$$y = \tfrac{1}{2}\left(\tfrac{z+1}{2^x} \dot{-} 1\right).$$

The minimalization of the primitive recursive predicate $\gamma(\cdot)$ above makes x the largest number such that $2^x|(z + 1)$. As shown in [5], for any number z, there are unique x and y such that $\langle x, y \rangle = z$ and the pairing function $\langle x, y \rangle$ is primitive recursive. For example, if $z = 27$, then

$$x = \min_{d \leq 28} \gamma(d) = 2;$$

$$y = \tfrac{1}{2}\left(\tfrac{28}{2^2} \dot{-} 1\right) = 3;$$

$$\langle 2, 3 \rangle = 2^2(2 \cdot 3 + 1) \dot{-} 1 = 27.$$

Let the functions $l(z)$ and $r(z)$ in (6) return the left and right components of any number z. Since the predicate $z = \langle x, y \rangle$ is primitive recursive, both

$$\begin{aligned} l(z) &= \min_{x \leq z}\{(\exists y)_{\leq z}\{z = \langle x, y \rangle\}\} \\ r(z) &= \min_{y \leq z}\{(\exists x)_{\leq z}\{z = \langle x, y \rangle\}\} \end{aligned} \tag{6}$$

are primitive recursive, because they are bounded minimalizations of the bounded existential quanitifcations of primitive recursive predicates. Thus, if $z = 27 = \langle 2, 3 \rangle$, then $l(z) = 2$, $r(z) = 3$, and, in general, $\langle l(z), r(z) \rangle = z$ for any number z.

Let p_n be the n-th prime so that $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, etc. Let $p_0 = 0$, by default. In [5], p_n is shown to be computed by a primitive recursive function, which we define as

$$\pi(i) = p_i. \tag{7}$$

Thus, $\pi(0) = 0$, $\pi(1) = 2$, $\pi(2) = 3$, $\pi(3) = 5$, $\pi(4) = 7$, $\pi(5) = 11$, etc.

Let $(a_1, \ldots, a_n)$ be a sequence of numbers. The function in (8) computes the Gödel number (G-number) of this sequence.

$$[a_1, \ldots, a_n] = \prod_{i=1}^{n} \pi(i)^{a_i} \tag{8}$$

The function $[a_1, \ldots, a_n]$ is shown to be primitive recursive in [5], because $x \cdot y$, x^y, $\pi(i)$ are primitive recursive, and a finite product of primitive recursive functions is, by

induction, primitive recursive. The G-number of the empty number sequence $()$ is 1. Thus, the G-number of $(3, 101, 7891, 1, 43)$ is

$$[3, 101, 7891, 1, 43] \quad = \quad 2^3 \cdot 3^{101} \cdot 5^{7891} \cdot 7^1 \cdot 11^{43}.$$

Let $x = [a_1, \ldots, a_n]$. The accessor function $(x)_i = a_i$ in (9), which returns the i-th element of x, is shown to be primitive recursive in [5].

$$(x)_i = \min_{t \leq x} \{ \pi(i)^{t+1} \nmid x \} \tag{9}$$

The function $(x)_i$ allows us to treat G-numbers as arrays (i.e., sequences of numbers). Thus, if $x = [1, 7, 13]$, then

$$\begin{array}{lcl} (x)_1 & = & 1; \\ (x)_2 & = & 7; \\ (x)_3 & = & 13; \\ (x)_j & = & 0, \text{if } j > 3. \end{array}$$

Since G-numbers are sequences, they have lengths. Thus, if $x = [a_1, a_2, \ldots, a_n]$, the length of x is the position of the last non-zero prime power in x and is computed by the primitive recursive function $\mathtt{Lt}(\cdot)$ in (10). We note that $\mathtt{Lt}([a_1, \ldots, a_n]) = n$ if and only if $a_n \neq 0$, and if $\mathtt{Lt}(x) = n$, then $[(x)_1, \ldots, (x)_n] = x$.

$$\mathtt{Lt}(x) = \min_{i \leq x} \{ (x)_i \neq 0 \wedge (\forall j)_{\leq x} \{ j \leq i \vee (x)_j = 0 \} \} \tag{10}$$

Thus,

$$\mathtt{Lt}(540) = \mathtt{Lt}([2, 3, 1]) = 3.$$

The function $\lfloor x/y \rfloor$ that returns the integer part of the quotient x/y is shown to be primitive recursive in [5]. Thus, $\lfloor 7/2 \rfloor = 3$; $\lfloor 2/5 \rfloor = 0$; $\lfloor 8/5 \rfloor = 1$, and $\lfloor x/0 \rfloor = 0$ for any number x.

3. G-Number Operators

In this section, we define several primitive recursive functions on G-numbers. We call these functions *G-number operators* and use them in Section 4 to show that several characteristics of chess and are pr-decidable and pr-computable.

3.1. Assignment

Let

$$P_0(i, b) \equiv 1 \leq i \leq \mathtt{Lt}(b) \wedge b \geq 1.$$

The function

$$\mathtt{set}(b, i, v) = \begin{cases} \left\lfloor \frac{b}{\pi(i)^{(b)_i}} \right\rfloor \cdot \pi(i)^v & \text{if } P_0(i, b), \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

assigns the value of the i-th element of b to v. By Theorem 5.4 (the Definition by Cases Theorem) of Chapter 3 in [5] (which we henceforth abbreviate as the DCT), $\mathtt{set}(\cdot)$ in (11) is primitive recursive, because $P_0(\cdot)$ is primitive recursive. Thus, if $b = [1, 2] = 2^1 3^2 = 18$, $i = 1$, and $v = 3$, then

$$\mathtt{set}([1, 2], 1, 3) = \left\lfloor \frac{b}{\pi(1)^{(b)_1}} \right\rfloor \cdot \pi(1)^3 =$$

$$\left\lfloor \frac{[1, 2]}{2^{([1,2])_1}} \right\rfloor \cdot 2^3 = \left\lfloor \frac{2^1 \cdot 3^2}{2^1} \right\rfloor \cdot 2^3 = [3, 2] = 72$$

or, more succinctly,

$$\texttt{set}(18,1,3) = 72.$$

If $b = 72$, $i = 2$, and $v = 4$, then

$$\texttt{set}(72,2,4) = 648,$$

because

$$\texttt{set}(\texttt{set}(18,1,3),2,4) = \texttt{set}(72,2,4) =$$

$$\left\lfloor \frac{72}{3^2} \right\rfloor \cdot 3^4 = [3,4] = 648.$$

If $b = [2,3,5,8,13]$ $i = 5$, $v = 21$, then

$$\texttt{set}(b,5,21) = \left\lfloor \frac{b}{11^{13}} \right\rfloor 11^{21} = [2,3,5,8,21] =$$

$$1439807679413191764324717997871250 0$$

or, more succinctly,

$$\texttt{set}(671680908528903807967125 00,5,21) = 1439807679413191764324717997871250 0.$$

3.2. Count

Let

$$\begin{array}{lcl} \texttt{cntx}(x,y,0) & = & 0, \\ \texttt{cntx}(x,y,t+1) & = & F_0(x,y,t,\texttt{cntx}(x,y,t)), \end{array}$$

where

$$F_0(x,y,t,c) = \begin{cases} 1+c & \text{if } (y)_{s(t)} = x, \\ c & \text{otherwise.} \end{cases}$$

The function $F_0(\cdot)$ is primitive recursive by the DCT. Therefore, the function $\texttt{cntx}(\cdot)$ is primitive recursive, because it is obtained from $s(x)$ and $F_0(\cdot)$ by composition and recursion. The function $\texttt{cnt}(\cdot)$ in (12) to compute the count of occurrences of x in y is also primitive, because it is obtained by composition from $\texttt{cntx}(\cdot)$ and $\texttt{Lt}(x)$.

$$\texttt{cnt}(x,y) = \texttt{cntx}(x,y,\texttt{Lt}(y)) \tag{12}$$

Let y = 30,870 = [1, 2, 1, 3]. Then

$$\begin{array}{lcl} \texttt{cnt}(1,y) & = & \texttt{cntx}(1,y,4) \\ & = & F_0(1,y,3,\texttt{cntx}(1,y,3)); \\ \texttt{cntx}(1,y,3) & = & F_0(1,y,2,\texttt{cntx}(1,y,2)); \\ \texttt{cntx}(1,y,2) & = & F_0(1,y,1,\texttt{cntx}(1,y,1)); \\ \texttt{cntx}(1,y,1) & = & F_0(1,y,0,\texttt{cntx}(1,y,0)); \\ \texttt{cntx}(1,y,0) & = & 0; \\ F_0(1,y,0,0) & = & 1, \text{since } (y)_{s(0)} = 1; \\ \texttt{cntx}(1,y,1) & = & F_0(1,y,0,0) = 1; \\ F_0(1,y,1,1) & = & 1, \text{since } (y)_{s(1)} \neq 1; \\ \texttt{cntx}(1,y,2) & = & F_0(1,y,1,1) = 1; \\ F_0(1,y,2,1) & = & 2, \text{since } (y)_{s(2)} = 1; \\ \texttt{cntx}(1,y,3) & = & F_0(1,y,2,1) = 2; \\ F_0(1,y,3,2) & = & 2, \text{since } (y)_{s(3)} \neq 1; \\ \texttt{cntx}(1,y,4) & = & F_0(1,y,3,2) = 2; \\ \texttt{count}(1,y) & = & 2. \end{array}$$

Analogously,

$$\begin{array}{l}\mathtt{cnt}(2,y)=\mathtt{cnt}(3,y)=1;\\ \mathtt{cnt}(10,y)=0.\end{array}$$

Let

$$\mathtt{in}(x,y)\equiv\mathtt{cnt}(x,y)\neq 0\equiv x\in y. \tag{13}$$

Thus,

$$\begin{array}{l}1\in 30870;\\ 2\in 30870;\\ 3\in 30870;\\ 10\notin 30870.\end{array}$$

Let $k\in y$ and let $P(x_1,\ldots,x_n)$, $n\geq 1$, be an n-ary predicate. We will adopt the following notational conventions for bounded quantifiers:

$$\begin{array}{rcl}(\forall k\in y)P(k,\ldots,x_n) & \equiv & (\forall t)_{\leq \mathtt{Lt}(y)}\{t<1\vee P((y)_t,\ldots,x_n)\};\\ (\exists k\in y)P(k,\ldots,x_n) & \equiv & (\exists t)_{\leq \mathtt{Lt}(y)}\{t>0\wedge P((y)_t,\ldots,x_n)\}.\end{array} \tag{14}$$

Thus, if $y=[1,2,3,4]$ and $P(x_1,x_2)\equiv x_1<x_2$, then

$$(\forall k\in y)\{P(k,5)\}=(\exists k\in y)\{P(k,2)\}=1,$$

whereas

$$(\forall k\in y)\{P(k,3)\}=(\exists k\in y)\{P(k,1)\}=0.$$

3.3. Append

The function

$$\mathtt{rap}(x,y)=\begin{cases}[x] & \text{if } \mathtt{Lt}(y)=0,\\ y\cdot\pi(\mathtt{Lt}(y)+1)^x & \text{otherwise}\end{cases} \tag{15}$$

appends x to the right of the rightmost element of y. By the DCT, the function $\mathtt{rap}(\cdot)$ in (15) is primitive recursive, because $x\cdot y$, $\mathtt{Lt}(y)$, x^y, $\pi(x)$, and x^y are. Thus,

$$\begin{array}{l}\mathtt{rap}(1,0)=\mathtt{rap}(1,1)=\pi(1)=[1];\\ \mathtt{rap}(1,[1])=\mathtt{rap}(1,2)=2\cdot\pi(\mathtt{Lt}(2)+1)^1=2\cdot\pi(2)^1=2\cdot 3^1=6=[1,1];\\ \mathtt{rap}(8,[2,3,5])=[2,3,5]\cdot\pi(4)^8=[2,3,5,8];\\ \mathtt{rap}(5,\mathtt{set}([0,3],1,2))=[2,3,5].\end{array}$$

More succinctly,

$$\begin{array}{l}\mathtt{rap}(1,0)=\mathtt{rap}(1,1)=2;\\ \mathtt{rap}(1,2)=6;\\ \mathtt{rap}(8,337500)=1945620337500;\\ \mathtt{rap}(5,\mathtt{set}(27,1,2))=\mathtt{rap}(5,108)=337500.\end{array}$$

3.4. Concatenation

Let

$$\begin{array}{rcl}\mathtt{lc}(x_1,x_2,0) & = & x_2,\\ \mathtt{lc}(x_1,x_2,t+1) & = & F_1(x_1,t,\mathtt{lc}(x_1,x_2,t)),\end{array}$$

where

$$F_1(x,t,y)=\mathtt{rap}((x)_{s(t)},y).$$

The function $\mathtt{lc}(\cdot)$ is primitive recursive by definition. The function

$$x_1 \otimes_l x_2 = \mathtt{lc}(x_1, x_2, \mathtt{Lt}(x_1)), \tag{16}$$

which is also primitive recursive by definition, places all numbers in x_2, in order, to the left of the first number, if there is one, in x_1. We refer to the binary function in (16) as *left concatenation*. Thus,

$$0 \otimes_l x = 1 \otimes_l x = x.$$

If $x_1 = 2 = [1]$ and $x_2 = 4 = [2]$, then

$$2 \otimes_l 4 = [1] \otimes_l [2] = [2, 1] = 12,$$

because

$$\begin{array}{rcl}
[1] \otimes_l [2] & = & \mathtt{lc}([1], [2], \mathtt{Lt}([1])) \\
& = & \mathtt{lc}([1], [2], 1) \\
& = & F_1([1], 0, \mathtt{lc}([1], [2], 0)) \\
& = & F_1([1], 0, [2])) \\
& = & \mathtt{rap}(([1])_{s(0)}, [2]) \\
& = & \mathtt{rap}(([1])_1, [2]) \\
& = & \mathtt{rap}(1, [2]) \\
& = & [2, 1].
\end{array}$$

We also have

$$[2, 3] \otimes_l [1] = [1, 2, 3].$$

To compute $[3, 4, 5] \otimes_l [1, 2]$, we compute

$$\begin{array}{lcl}
\mathtt{lc}([3, 4, 5], [1, 2], 1) & = & \\
F_1([3, 4, 5], 0, \mathtt{lc}([3, 4, 5], [1, 2], 0) & = & \\
F_1([3, 4, 5], 0, [1, 2]) & = & \\
\mathtt{rap}(([3, 4, 5])_{s(0)}, [1, 2]) & = & [1, 2, 3]
\end{array}$$

and, hence,

$$\begin{array}{lcl}
\mathtt{lc}([3, 4, 5], [1, 2], 2) & = & \\
F_1([3, 4, 5], 1, \mathtt{lc}([3, 4, 5], [1, 2], 1) & = & \\
F_1([3, 4, 5], 1, [1, 2, 3]) & = & \\
\mathtt{rap}(([3, 4, 5])_{s(1)}, [1, 2, 3]) & = & [1, 2, 3, 4],
\end{array}$$

which gives us

$$\begin{array}{rcl}
[3, 4, 5] \otimes_l [1, 2] & = & \mathtt{lc}([3, 4, 5], [1, 2], \mathtt{Lt}([3, 4, 5])) \\
& = & \mathtt{lc}([3, 4, 5], [1, 2], 3) \\
& = & F_1([3, 4, 5], 2, \mathtt{lc}([3, 4, 5], [1, 2], 2)) \\
& = & F_1([3, 4, 5], 2, [1, 2, 3, 4]) \\
& = & \mathtt{rap}(([3, 4, 5])_{s(2)}, [1, 2, 3, 4]) \\
& = & [1, 2, 3, 4, 5]
\end{array}$$

or, more succinctly,

$$2025000 \otimes_l 18 = 870037764750.$$

Let

$$F_2(x) = \begin{cases} 0 & \text{if } \mathtt{Lt}(x) = 0, \\ [(x)_1] & \text{if } \mathtt{Lt}(x) = 1, \\ F_3(x, \mathtt{Lt}(x) \dot{-} 2) & \text{if } \mathtt{Lt}(x) > 1, \end{cases}$$

where

$$\begin{array}{lcl} F_3(x,0) & = & (x)_{\mathtt{Lt}(x)\dot{-}1} \otimes_l (x)_{\mathtt{Lt}(x)}, \\ F_3(x,t+1) & = & (x)_{s(t)} \otimes_l F_3(x,t). \end{array}$$

Since $F_3(\cdot)$ is primitive recursive by definition and $F_2(\cdot)$ is primitive recursive by the DCT, the left concatenation of $x_1, \ldots, x_k$, $k \geq 1$, in (17) is primitive recursive.

$$\otimes_l |_{i=1}^{k} x_i = F_2([x_1, \ldots, x_k]) \tag{17}$$

Thus, if $x_1 = [3,4,5]$, $x_2 = [1,2]$, $x_3 = [10,11]$, then

$$\begin{array}{lcl} \otimes_l|_{i=1}^{3} x_i & = & F_2([x_1,x_2,x_3]) \\ & = & F_3([x_1,x_2,x_3],1) \\ & = & x_1 \otimes_l F_3([x_1,x_2,x_3],0) \\ & = & x_1 \otimes_l (x_2 \otimes_l x_3) \\ & = & x_1 \otimes_l [10,11,1,2] \\ & = & [10,11,1,2,3,4,5] \\ & = & 23988103528747128574003 20. \end{array}$$

In general,

$$\begin{array}{l} \otimes_l|_{i=1}^{k} x_i = x_1 \otimes_l x_2 \otimes_l \ldots \otimes_l x_k = \\ (\ldots((x_1 \otimes_l x_2) \otimes_l \ldots \otimes_l x_k)\ldots) = \\ (\ldots(x_1 \otimes_l (x_2 \otimes_l (\ldots \otimes_l (x_{k-1} \otimes_l x_k)\ldots)))\ldots). \end{array} \tag{18}$$

The primitive recursive function to do the *right concatenation* of x_1 and x_2 (i.e., the operation of placing all numbers of x_2, in order, to the right of the rightmost number in x_1) is defined in (19).

$$x_1 \otimes_r x_2 = x_2 \otimes_l x_1 \tag{19}$$

Thus,

$$\begin{array}{l} [1] \otimes_r [2] = [2] \otimes_l [1] = [1,2]; \\ [2,3] \otimes_r [1] = [1] \otimes_l [2,3] = [2,3,1]; \\ [3,4,5] \otimes_r [1,2] = [1,2] \otimes_l [3,4,5] = [3,4,5,1,2]. \end{array}$$

The right concatenation of any numbers $x_1, \ldots, x_k$ in (20) is primitive recursive, because it is a composition of two primitive recursive functions.

$$\otimes_r |_{i=1}^{k} x_i = F_2([x_k, \ldots, x_1]) \tag{20}$$

Let $x_1 = [3,4,5]$, $x_2 = [1,2]$, $x_3 = [10,11]$. Then

$$\begin{array}{lcl} \otimes_r|_{i=1}^{3} x_i & = & F_2([x_3,x_2,x_1]) \\ & = & F_3([x_3,x_2,x_1],1) \\ & = & x_3 \otimes_l F_3([x_3,x_2,x_1],0) \\ & = & x_3 \otimes_l (x_2 \otimes_l x_1) \\ & = & x_3 \otimes_l [3,4,5,1,2] \\ & = & [3,4,5,1,2,10,11]. \end{array}$$

In general,

$$\begin{array}{l} \otimes_r|_{i=1}^{k} x_i = x_1 \otimes_r x_2 \otimes_r \ldots \otimes_r x_k = \\ (\ldots((x_1 \otimes_r x_2) \otimes_r \ldots \otimes_r x_k)\ldots) = \\ (\ldots(x_1 \otimes_r (x_2 \otimes_r (\ldots \otimes_r (x_{k-1} \otimes_r x_k)\ldots)))\ldots). \end{array} \tag{21}$$

We will use the notation $\otimes_r|_{i\in g}$, for G-number g, to denote the right concatenation of the elements indexed from $(g)_1$ to $(g)_{\mathtt{Lt}(g)}$ in increments of 1. Thus, if $g = [1,2,3]$ and $x_1 = [3,4,5], x_2 = [1,2], x_3 = [10,11]$, then

$$\otimes_l|_{i\in g} x_i = \otimes_l|_{i=1}^{3} x_i = [10,11,1,2,3,4,5];$$

$$\otimes_r|_{i\in g} x_i = \otimes_r|_{i=1}^{3} x_i = [3,4,5,1,2,10,11].$$

3.5. G-Number Generator

Let

$$\mathtt{gnx}(l,u,k,0) = [l],$$
$$\mathtt{gnx}(l,u,k,t+1) = \mathtt{gnnx}(l,u,k,\mathtt{gnx}(l,u,k,t),t),$$

where

$$\mathtt{gnnx}(l,u,k,z,t) = \begin{cases} z \otimes_r [l+s(t)k] & \text{if } l+s(t)k \le u, \\ z & \text{otherwise.} \end{cases}$$

Since $\mathtt{gnnx}(\cdot)$ is primitive recursive by the DCT, so is $\mathtt{gnx}(\cdot)$. Let

$$\mathtt{ggn}(l,u,k) = \begin{cases} \mathtt{gnx}(l,u,k,s(u \dot{-} l)) & \text{if } k > 0 \wedge l \le u, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

The function $\mathtt{ggn}(\cdot)$ is primitive recursive by the DCT. Thus, if we want to generate the G-number whose elements are in the range from 5 to 7 and the range is iterated through in increments of 1, we have

$$\begin{aligned}
\mathtt{ggn}(5,7,1) &= \mathtt{gnx}(5,7,1,3) \\
&= \mathtt{gnnx}(5,7,1,\mathtt{gnx}(5,7,1,2),2); \\
\mathtt{gnx}(5,7,1,2) &= \mathtt{gnnx}(5,7,1,\mathtt{gnx}(5,7,1,1),1); \\
\mathtt{gnx}(5,7,1,1) &= \mathtt{gnnx}(5,7,1,\mathtt{gnx}(5,7,1,0),0); \\
\mathtt{gnx}(5,7,1,0) &= [5]; \\
\mathtt{gnnx}(5,7,1,\mathtt{gnx}(5,7,1,0),0) &= \mathtt{gnnx}(5,7,1,[5],0) \\
&= [5] \otimes_r [5+1\cdot 1] \\
&= [5] \otimes_r [6] = [5,6]; \\
\mathtt{gnx}(5,7,1,1) &= [5,6]; \\
\mathtt{gnx}(5,7,1,2) &= \mathtt{gnnx}(5,7,1,[5,6],1) \\
&= [5,6] \otimes_r [5+2\cdot 1] \\
&= [5,6] \otimes_r [7] \\
&= [5,6,7]; \\
\mathtt{gnnx}(5,7,1,[5,6,7],2) &= [5,6,7]; \\
\mathtt{gnx}(5,7,1,3) &= [5,6,7]; \\
\mathtt{ggn}(5,7,1) &= [5,6,7].
\end{aligned}$$

If we want to generate the G-number whose elements are in the range from 5 to 7 iterated through in increments of 2, we have

$$\begin{array}{lcl}
\texttt{ggn}(5,7,2) & = & \texttt{gnx}(5,7,2,3) \\
 & = & \texttt{gnnx}(5,7,2,\texttt{gnx}(5,7,2,2),2); \\
\texttt{gnx}(5,7,2,2) & = & \texttt{gnnx}(5,7,2,\texttt{gnx}(5,7,2,1),1); \\
\texttt{gnx}(5,7,2,1) & = & \texttt{gnnx}(5,7,2,\texttt{gnx}(5,7,2,0),0); \\
\texttt{gnx}(5,7,2,0) & = & [5]; \\
\texttt{gnnx}(5,7,2,\texttt{gnx}(5,7,2,0),0) & = & \texttt{gnnx}(5,7,2,[5],0) \\
 & = & [5] \otimes_r [5+1\cdot 2] \\
 & = & [5] \otimes_r [7] = [5,7]; \\
\texttt{gnx}(5,7,2,1) & = & [5,7]; \\
\texttt{gnx}(5,7,2,2) & = & \texttt{gnnx}(5,7,2,[5,7],1) \\
 & = & [5,7]; \\
\texttt{gnnx}(5,7,2,[5,7],2) & = & [5,7]; \\
\texttt{gnx}(5,7,2,3) & = & [5,7]; \\
\texttt{ggn}(5,7,2) & = & [5,7].
\end{array}$$

We observe that $\texttt{ggn}(5,7,k) = 0$ for any $k > 2$. If $u = l$ and $k > 0$, then $\texttt{ggn}(l,u,k) = [l]$.

3.6. Subsequence

The function $\texttt{ssq}(\cdot)$ in (23) returns the subsequence of G-number z specified by its start and end positions i and j in z.

$$\texttt{ssq}(z,i,j) = \begin{cases} \prod_{k=i}^{j} \pi(k)^{(z)_k} & \text{if } 0 < i \le j \le \texttt{Lt}(z), \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Thus, if $z = [1,2,3,4,5]$, then

$$\texttt{ssq}(z,1,3) = \prod_{k=1}^{3} \pi(k)^{(z)_k} = [1,2,3]$$

and

$$\texttt{ssq}(z,4,5) = \prod_{k=4}^{5} \pi(k)^{(z)_k} = [4,5].$$

3.7. Removal

Let

$$\begin{array}{lcl}
\texttt{rmx}(x,y,0) & = & [\,], \\
\texttt{rmx}(x,y,t+1) & = & F_4(x,y,\texttt{rmx}(x,y,t),s(t)),
\end{array}$$

where

$$F_4(x,y,z,i) = \begin{cases} z & \text{if } (y)_i = x, \\ [(y)_i] \otimes_l z & \text{otherwise.} \end{cases}$$

Since $F_4(\cdot)$ is primitive by the DCT, $\texttt{rmx}(\cdot)$ is also primitive recursive. The primitive recursive function $\texttt{rm}(x,y)$ in (24) removes all occurrences of x from y.

$$\texttt{rm}(x,y) = \texttt{rmx}(x,y,\texttt{Lt}(y)) \tag{24}$$

Let $y = [2,3,1]$. Then

$$\begin{aligned}
\mathtt{rm}(1,y) &= \mathtt{rmx}(1,y,3)\\
&= F_4(1,y,\mathtt{rmx}(1,y,2),3)\\
&= F_4(1,y,F_4(1,y,\mathtt{rmx}(1,y,1),2),3)\\
&= F_4(1,y,F_4(1,y,F_4(1,y,\mathtt{rmx}(1,y,0),1),2),3)\\
&= F_4(1,y,F_4(1,y,F_4(1,y,[\,],1),2),3)\\
&= F_4(1,y,F_4(1,y,[2],2),3)\\
&= F_4(1,y,[2,3],3)\\
&= [2,3].
\end{aligned}$$

Let $y = [5,1,2,3,2]$. Then $\mathtt{rm}(2,y) = [5,1,3]$.

Sometimes it is required to remove all duplicate elements from a G-number. Let

$$\begin{aligned}
\mathtt{rmdx}(y,0) &= [\,],\\
\mathtt{rmdx}(y,t+1) &= F_5(y,\mathtt{rmdx}(y,t),s(t)),
\end{aligned}$$

where

$$F_5(y,z,i) = \begin{cases} z \otimes_r [(y)_i] & \text{if } (y)_i \notin z,\\ z & \text{otherwise.}\end{cases}$$

Since $F_5(\cdot)$ is primitive recursive by the DCT, so is $\mathtt{rmdx}(\cdot)$. The primitive recursive function $\mathtt{rmd}(x,y)$ in (25) removes all duplicates from y.

$$\mathtt{rmd}(y) = \mathtt{rmdx}(y,\mathtt{Lt}(y)) \tag{25}$$

Let $y = [5,3,5]$. Then

$$\begin{aligned}
\mathtt{rmd}(y) &= \mathtt{rmdx}(y,3)\\
&= F_5(y,\mathtt{rmdx}(y,2),3)\\
&= F_5(y,F_5(y,\mathtt{rmdx}(y,1),2),3)\\
&= F_5(y,F_5(y,F_5(y,\mathtt{rmdx}(y,0),1),2),3)\\
&= F_5(y,F_5(y,F_5(y,[\,],1),2),3)\\
&= F_5(y,F_5(y,[5],2),3)\\
&= F_5(y,[5,3],3)\\
&= [5,3].
\end{aligned}$$

3.8. Predicate Mapping

We define a function that takes a unary primitive recursive predicate $P(x)$ and G-number y and returns another G-number y' such that the elements of y' are the elements of y that satisfy the predicate. Let $P(x)$ be a primitive recursive predicate and let

$$\begin{aligned}
\mathtt{mapx}_P(y,0) &= [\,],\\
\mathtt{mapx}_P(y,t+1) &= F_6(y,\mathtt{mapx}_P(y,t),s(t)),
\end{aligned}$$

where

$$F_6(y,z,i) = \begin{cases} z & \text{if } \neg P((y)_i),\\ [(y)_i] \otimes_l z & \text{if } P((y)_i),\end{cases}$$

and let

$$\mathtt{map}_P(y) = \mathtt{mapx}_P(y,\mathtt{Lt}(y)). \tag{26}$$

Since $P(x)$ is primitive recursive, $F_6(\cdot)$ is primitive recursive by the DCT. Thus, $\mathtt{map}_P(\cdot)$ is primitive recursive by definition for any unary primitive recursive predicate $P(x)$.

Let

$$\mathtt{prime}(x) \equiv x > 1 \ \wedge \ (\forall t)_{\le x}\{t = 1 \vee t = x \vee t \nmid x\}, \tag{27}$$

which is shown to be primitive recursive in [5]. Let $P(x) \equiv \mathtt{prime}(x)$ and $y = [5,6,7]$. Then

$$\begin{aligned}
\mathtt{map}_P(y) &= \mathtt{mapx}_P(y, \mathtt{Lt}(y)) \\
&= \mathtt{mapx}_P(y, 3) \\
&= F_6(y, \mathtt{mapx}_P(y, 2), 3) \\
&= F_6(y, F_6(y, \mathtt{mapx}_P(y, 1), 2), 3) \\
&= F_6(y, F_6(y, F_6(y, \mathtt{mapx}_P(y, 0), 1), 2), 3) \\
&= F_6(y, F_6(y, F_6(y, [\,], 1), 2), 3) \\
&= F_6(y, F_6(y, [5], 2), 3) \\
&= F_6(y, [5], 3) \\
&= [5, 7].
\end{aligned}$$

As a notational shortcut, we will occasionally write the actual predicate in the subscript to $\mathtt{map}(\cdot)$ (e.g., $\mathtt{map}_{\mathtt{prime}(x)}(y)$). Thus, we can summarize the above example as

$$\mathtt{map}_P(y) = \mathtt{map}_{\mathtt{prime}(x)}(y) = [5, 7].$$

Similarly,

$$\mathtt{map}_{x>5}(y) = [6, 7].$$

If $P(x)$ is a primitive recursive predicate, then so is $\neg P(x)$. Consequently, $\mathtt{map}_{\neg P}(y)$ is the G-number that consists of all numbers x of y for which $\neg P(x)$ is true. Then the primitive recursive function

$$\mathtt{rmp}_P(y) = \mathtt{map}_{\neg P}(y) \tag{28}$$

removes from G-number y all elements $x \in y$ for which $P(x)$ holds. Thus, if $P(x) \equiv \mathtt{prime}(x)$ and $y = [3, 5, 6, 7]$, then

$$\mathtt{rmp}_P(y) = \mathtt{map}_{\neg \mathtt{prime}}(y) = [6]$$

and

$$\mathtt{rmp}_{x \geq 5}(y) = \mathtt{map}_{x<5}(y) = [3].$$

3.9. Position

Let

$$\begin{aligned}
\mathtt{psx}(x, y, 0) &= [\,], \\
\mathtt{psx}(x, y, t+1) &= F_7(x, y, \mathtt{psx}(x, y, t), s(t)),
\end{aligned}$$

where

$$F_7(x, y, z, i) = \begin{cases} [i] \otimes_l z & \text{if } (y)_i = x, \\ z & \text{otherwise.} \end{cases}$$

Since $(y)_i$ is primitive recursive, $F_7(\cdot)$ is primitive recursive by the DCT. Thus, $\mathtt{psx}(\cdot)$ is primitive recursive by definition. The primitive recursive function

$$\mathtt{psn}(x, y) = \mathtt{psx}(x, y, \mathtt{Lt}(y)) \tag{29}$$

computes the G-number of all positions of x in y. Let $x = 1$, $y = [1, 3, 1]$. Then

$$\begin{aligned}
\mathtt{psn}(1, y) &= \mathtt{psx}(1, y, \mathtt{Lt}(y)) \\
&= \mathtt{psx}(1, y, 3) \\
&= F_7(1, y, \mathtt{psx}(1, y, 2), 3) \\
&= F_7(1, y, F_7(1, y, \mathtt{psx}(1, y, 1), 2), 3) \\
&= F_7(1, y, F_7(1, y, F_7(1, y, \mathtt{psx}(1, y, 0), 1), 2), 3) \\
&= F_7(1, y, F_7(1, y, F_7(1, y, [\,], 1), 2), 3) \\
&= F_7(1, y, F_7(1, y, [1], 2), 3) \\
&= F_7(1, y, [1], 3) \\
&= [1, 3].
\end{aligned}$$

In general,

$$(\forall t \in \mathtt{psn}(x,y))\{(y)_t = x\}.$$

3.10. Association

Let the primitive recursive function

$$\mathtt{asx}(x,y) = \min_{t \le Lt(y)} \{t > 0 \wedge x = l((y)_t)\} \tag{30}$$

return the smallest index t of $\langle i,j\rangle \in y$ such that $x = i$. Thus, if

$$y = [\langle 10,100\rangle, \langle 20,200\rangle, \langle 30,300\rangle],$$

then

$$\begin{array}{rcl} \mathtt{asx}(10,y) & = & 1; \\ \mathtt{asx}(20,y) & = & 2; \\ \mathtt{asx}(30,y) & = & 3. \end{array}$$

We define the primitive recursive function

$$\mathtt{asc}(x,y) = (y)_{\mathtt{asx}(x,y)} \tag{31}$$

to return the pair from y at the index t returned by $\mathtt{asx}(\cdot)$. Thus, if

$$y = [\langle 10,100\rangle, \langle 20,200\rangle, \langle 30,300\rangle],$$

then

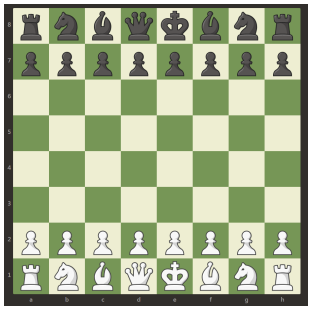
$$\begin{array}{rcccccl} \mathtt{asc}(10,y) & = & (y)_{\mathtt{asx}(10,y)} & = & (y)_1 & = & \langle 10,100\rangle; \\ \mathtt{asc}(20,y) & = & (y)_{\mathtt{asx}(20,y)} & = & (y)_2 & = & \langle 20,200\rangle; \\ \mathtt{asc}(30,y) & = & (y)_{\mathtt{asx}(30,y)} & = & (y)_3 & = & \langle 30,200\rangle; \\ \mathtt{asc}(13,y) & = & (y)_{\mathtt{asx}(13,y)} & = & (y)_0 & = & 0. \end{array}$$

4. Chess

4.1. Boards

Figure 1 shows the starting board of any chess game. It consists of 64 cells where each of the two players (the white and the black), has 16 pieces. We encode this board as G-number b of 64 numbers, each of which encodes the contents of the corresponding cell. This number b can, when convenient for visualization, be construed as a matrix shown below the board picture in Figure 1.

An empty cell is encoded as 1. A white piece on the starting board is encoded as a number from 2 up to 17. Thus, west to east, the white pawns are encoded as 2, 3, 4, 5, 6, 7, 8, 9, the white rooks as 10 (west) and 17 (east), the white knights as 11 (west) and 16 (east), the white bishops as 12 (west or dark-colored) and 15 (east or light-colored), the white queen as 13, and the white king as 14.



26	27	28	29	30	31	32	33
18	19	20	21	22	23	24	25
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17

Figure 1. Starting chess board (**above**) encoded as G-number b (**below**).

A black piece on the starting board is encoded as a number from 18 up to 33. Thus, west to east, the black pawns are encoded as 18, 19, 20, 21, 22, 23, 24, 25, the black rooks as 26 (west) and 33 (east), the black knights as 27 (west) and 32 (east), the black bishops as 28 (west or light-colored) and 31 (east or dark-colored), the black queen as 29, and the black king as 30.

Each cell on board b has a unique position shown in Figure 2. Let

$$\zeta_{ps} \quad = \quad \mathtt{ggn}(1, 64, 1) \tag{32}$$

denote the G-number of all positions from 1 to 64. A position k is a *valid position* if $k \in \zeta_{ps}$. Thus, if b is the board in Figure 1, then

$$\begin{array}{lcl} (b)_1 & = & 26; \\ (b)_2 & = & 27; \\ (b)_{63} & = & 16; \\ (b)_{64} & = & 17; \\ (b)_i & = & 0, i < 1 \text{ or } i > 64; \end{array}$$

and

$$(\forall i \in \mathtt{ggn}(17, 48))\{(b)_i = 1\}.$$

Since there are exactly 32 empty cells, we have

$$\mathtt{cnt}(1, b) = 32.$$

The position of the black and white kings on any valid board b (valid boards are defined below) are

$$\begin{array}{l} (\mathtt{psn}(14, b))_1; \\ (\mathtt{psn}(30, b))_1. \end{array}$$

Thus, for b in Figure 1, we have

$$\begin{array}{l} (\mathtt{psn}(14, b))_1 = 61; \\ (\mathtt{psn}(30, b))_1 = 5. \end{array}$$

Figure 3 shows board b' that is the result of two moves made by each player on b in Figure 1. Thus, the following predicates are true:

$$\begin{array}{llllllll}
(b)_{53} = 6 & \wedge & (b)_{37} = 1 & \wedge & (b')_{53} = 1 & \wedge & (b')_{37} = 6; \\
(b)_{63} = 16 & \wedge & (b)_{46} = 1 & \wedge & (b')_{63} = 1 & \wedge & (b')_{46} = 16; \\
(b)_{13} = 22 & \wedge & (b)_{29} = 1 & \wedge & (b')_{13} = 1 & \wedge & (b')_{29} = 22; \\
(b)_{7} = 32 & \wedge & (b)_{22} = 1 & \wedge & (b')_{7} = 1 & \wedge & (b')_{22} = 32.
\end{array}$$

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Figure 2. Positions in chess board b.

4.2. Pawn Metamorphosis

If a pawn reaches the first row on the opposite side of the board, it transforms into a queen, a rook, a bishop, or a king of the same color. We call these positions *metamorphic positions* and give them in Table 1.

Let $2 \leq j \leq 8$ be a white pawn and let k be one of its metamorphic positions. If j becomes a queen, the new white queen is encoded as $\langle\langle j,k\rangle,37\rangle$; if j becomes a rook, the new white rook is encoded as $\langle\langle j,k\rangle,41\rangle$; if j becomes a bishop, the new white bishop is encoded as $\langle\langle j,k\rangle,43\rangle$; if j becomes a knight, the new white knight is encoded as $\langle\langle j,k\rangle,47\rangle$.

Let $18 \leq j \leq 25$ be a black pawn and let k be one of its metamorphic positions. If j becomes a queen, the new black queen is encoded as $\langle\langle j,k\rangle,53\rangle$; if j becomes a rook, the new black rook is encoded as $\langle\langle j,k\rangle,59\rangle$; if j becomes a bishop, the new black bishop is encoded as $\langle\langle j,k\rangle,61\rangle$; if j becomes a knight, the new black knight is encoded as $\langle\langle j,k\rangle,67\rangle$.

26	27	28	29	30	31	1	33
18	19	20	21	1	23	24	25
1	1	1	1	1	32	1	1
1	1	1	1	22	1	1	1
1	1	1	1	6	1	1	1
1	1	1	1	1	16	1	1
2	3	4	5	1	7	8	9
10	11	12	13	14	15	1	17

Figure 3. Board b' after two moves (i.e., (1) e4 e5 and (2) Kf3 Kf6) by each player on b in Figure 1.

Table 1. Metamorphic positions for the white and black pawns encoded as G-numbers and the symbols that denote the G-numbers.

Pawn	Metamorphic Positions
2	[1, 2, 3, 4, 5, 6, 7]
3	[1, 2, 3, 4, 5, 6, 7, 8]
4	[1, 2, 3, 4, 5, 6, 7, 8]
5	[1, 2, 3, 4, 5, 6, 7, 8]
6	[1, 2, 3, 4, 5, 6, 7, 8]
7	[1, 2, 3, 4, 5, 6, 7, 8]
8	[1, 2, 3, 4, 5, 6, 7, 8]
9	[2, 3, 4, 5, 6, 7, 8]
18	[57, 58, 59, 60, 61, 62, 63]
19	[57, 58, 59, 60, 61, 62, 63, 64]
20	[57, 58, 59, 60, 61, 62, 63, 64]
21	[57, 58, 59, 60, 61, 62, 63, 64]
22	[57, 58, 59, 60, 61, 62, 63, 64]
23	[57, 58, 59, 60, 61, 62, 63, 64]
24	[57, 58, 59, 60, 61, 62, 63, 64]
25	[58, 59, 60, 61, 62, 63, 64]

Thus, if white pawn 2 becomes a queen at position 7, the new white queen is $\langle\langle 2,7\rangle,37\rangle$; if 2 becomes a rook at position 7, the new white rook is $\langle\langle 2,7\rangle,41\rangle$; if 2 becomes a bishop at position 7, the new white bishop is $\langle\langle 2,7\rangle,43\rangle$; if 2 becomes a knight, the new white knight is $\langle\langle 2,7\rangle,47\rangle$.

Similarly, if black pawn 19 becomes a queen at position 59, the new black queen is $\langle\langle 19,59\rangle,53\rangle$; if 19 becomes a rook at position 59, the new black rook is $\langle\langle 19,59\rangle,59\rangle$; if 19 becomes a bishop at position 59, the new black bishop is $\langle\langle 19,59\rangle,61\rangle$; if 19 becomes a knight at position 59, the new black knight is $\langle\langle 19,59\rangle,67\rangle$.

Since for any z, $z = \langle x,y\rangle$, where x and y are unique (See Equation (5)), this encoding scheme for pawn metamorphosis ensures that every new piece is encoded as a unique number. Each player can obtain at most eight new pieces through pawn metamorphosis. The supplementary materials contain the encoding tables for all pieces that can be obtained through pawn metamorphosis.

We call the pieces into which the pawns transform at the metamorphic positions *metamorphic pieces* and refer to the pieces on the original board as *regular pieces*. Formally, j is a regular white piece if $j \in \mathtt{ggn}(2,17,1)$, is a regular black piece if $j \in \mathtt{ggn}(18,33,1)$, and is a regular piece if j is a regular white or black piece.

Tables S1 and S2 in the Supplementary Materials contain all unique numbers of the white and black pieces under this encoding scheme that can be obtained through pawn metamorphosis with the primitive recursive pairing function. Let

$$\begin{aligned}\zeta_{\mu}^{w} &= [\langle\langle 2,1\rangle,37\rangle,\ldots,\langle\langle 9,8\rangle,47\rangle];\\ \zeta_{\mu}^{b} &= [\langle\langle 18,57\rangle,53\rangle,\ldots,\langle\langle 25,64\rangle,67\rangle]\end{aligned} \tag{33}$$

be the G-numbers of the white, black, and all metamorphic pieces.

4.3. Valid Pieces and Boards

Let the primitive recursive predicates

$$\begin{array}{rcl} \mathtt{wp}(j) & \equiv & \{2 \leq j \leq 17\} \vee \{j \in \zeta_{\mu}^{w}\}; \\ \mathtt{bp}(j) & \equiv & \{18 \leq j \leq 33\} \vee \{j \in \zeta_{\mu}^{b}\} \end{array} \tag{34}$$

hold when j is a valid white or black piece, respectively. A chess piece j is *valid* if and only if the primitive recursive predicate

$$\mathtt{vlp}(j) \equiv \mathtt{wp}(j) \vee \mathtt{bp}(j).$$

holds (i.e., $\mathtt{vlp}(j) = 1$). Let

$$\begin{array}{rcl} \zeta_{wp} & = & \mathtt{ggn}(2, 17, 1) \otimes_r \zeta_{\mu}^{w}; \\ \zeta_{bp} & = & \mathtt{ggn}(18, 33, 1) \otimes_r \zeta_{\mu}^{b}; \\ \zeta_{cp} & = & \zeta_{wp} \otimes_r \zeta_{bp} \end{array} \tag{35}$$

denote the G-numbers of the white pieces (ζ_{wp}), the black pieces (ζ_{bp}), and all chess pieces (ζ_{cp}).

Let us consider if the *valid chess board* characteristic is pr-decidable. If b is a valid chess board, it must contain exactly one white king (14) and exactly one black king (30). The primitive recursive predicate

$$P_1(b) \equiv \mathtt{cnt}(14, b) = \mathtt{cnt}(30, b) = 1$$

holds when b has this characteristic.

Board b is valid if the number of occurrences of each regular or metamorphic piece on b, with the exception of the two kings, is 0 (if the piece is captured) or exactly 1 (if the piece is not captured). The primitive recursive predicate

$$P_2(b) \equiv (\forall i \in \zeta_{cp})\{\{i = 14 \vee i = 30\} \vee \mathtt{cnt}(i, b) \leq 1\}$$

ensures that the number of occurrences for each piece on b, regular or metamorphic, so long as that piece is not a king, is 0 or 1.

If b is valid, then the count of the empty cells on it is at least 32 (i.e., b is the starting board) and at most 62 (i.e., when only the kings are present). The primitive recursive predicate

$$P_3(b) \equiv 32 \leq \mathtt{cnt}(1, b) \leq 62$$

holds when the count of the empty cells on b is between 32 and 62.

If board b is valid, then the regular and metamorphic black and white bishops, when present on b, must be in the appropriate light-colored and dark-colored positions. Tables S3–S6 in the Supplementary Materials contain the encoding tables for all metamorphic light-colored and dark-colored bishops.

Let $\zeta_{\mu}^{lc,wb}$ denote the G-number of the numbers of the metamorphic light-colored white bishops in Table S3, $\zeta_{\mu}^{dc,wb}$—the G-number of the numbers of the metamorphic dark-colored white bishops in Table S4, $\zeta_{\mu}^{lc,bb}$—the G-number of the numbers of the metamorphic light-colored black bishops in Table S5, $\zeta_{\mu}^{dc,bb}$—the G-number of the numbers of the metamorphic dark-colored black bishops in Table S6. Let

$$\begin{array}{rcl} \zeta_{\mu}^{lcb} & = & \zeta_{\mu}^{lc,wb} \oplus_r \zeta_{\mu}^{lc,bb}; \\ \zeta_{\mu}^{dcb} & = & \zeta_{\mu}^{dc,wb} \oplus_r \zeta_{\mu}^{dc,bb} \end{array} \tag{36}$$

be the G-numbers of the light-colored and dark-colored metamorphic bishops.

The primitive recursive predicates

$$\begin{aligned} \mathtt{wk}_\mu(j) &\equiv j \in \zeta_\mu^w \wedge r(j) = 47; \\ \mathtt{bk}_\mu(j) &\equiv j \in \zeta_\mu^b \wedge r(j) = 67 \end{aligned} \tag{37}$$

hold if j is a metamorphic white or black knight, respectively. The primitive recursive predicates

$$\begin{aligned} \mathtt{wlb}_\mu(j) &\equiv j \in \zeta_\mu^{lcb} \wedge r(j) = 43; \\ \mathtt{blb}_\mu(j) &\equiv j \in \zeta_\mu^{lcb} \wedge r(j) = 61 \end{aligned} \tag{38}$$

hold if j is a metamorphic white or black light-colored bishop, respectively. The primitive recursive predicates

$$\begin{aligned} \mathtt{wdb}_\mu(j) &\equiv j \in \zeta_\mu^{dcb} \wedge r(j) = 43; \\ \mathtt{bdb}_\mu(j) &\equiv j \in \zeta_\mu^{dcb} \wedge r(j) = 61; \end{aligned} \tag{39}$$

hold if j is a metamorphic white or black dark-colored knight, respectively. The primitive recursive predicates

$$\begin{aligned} \mathtt{kn}_\mu(j) &\equiv \mathtt{wk}_\mu(j) \vee \mathtt{bk}_\mu(j); \\ \mathtt{bh}_\mu(j) &\equiv \mathtt{wdb}_\mu(j) \vee \mathtt{bdb}_\mu(j) \end{aligned} \tag{40}$$

hold if j is a metamorphic knight or bishop, respectively.

If board b is valid, then the regular light-colored bishops 15 and 28 and any metamorphic light-colored bishops, when present, must be located on the light-colored positions and the regular dark-color bishops 12 and 31 and any metamorphic dark-colored bishops, when present, must be located on the dark-colored positions. Let

$$\begin{aligned} \zeta_{lps} = [&1, 3, 5, 7, 10, 12, 14, 16, 17, 19, 21, 23, 26, 28, 30, 32, 33, \\ &35, 37, 39, 42, 44, 46, 48, 49, 51, 53, 55, 58, 60, 62, 64] \end{aligned} \tag{41}$$

be the G-number of all light-colored positions on b (See Figure 4) and let

$$\begin{aligned} \zeta_{dps} = [&2, 4, 6, 8, 9, 11, 13, 15, 18, 20, 22, 24, 25, 27, 29, 31, 34, \\ &36, 38, 40, 41, 43, 45, 47, 50, 52, 54, 56, 57, 59, 61, 63] \end{aligned} \tag{42}$$

be the G-number of all dark-colored positions on b (See Figure 5). The primitive recursive predicate

$$\begin{aligned} P_4(b) \equiv\ & \{(\forall j \in [15, 28] \oplus_r \zeta_\mu^{lcb})\{j \notin b \vee (\mathtt{psn}(j, b))_1 \in \zeta_{lps}\} \wedge \\ & \{(\forall j \in [12, 31] \oplus_r \zeta_\mu^{dcb})\{j \notin b \vee (\mathtt{psn}(j, b))_1 \in \zeta_{dps}\} \end{aligned}$$

holds of b when all bishops, if present, are in the positions of the appropriate color.

1	0	**3**	0	**5**	0	**7**	0
0	**10**	0	**12**	0	**14**	0	**16**
17	0	**19**	0	**21**	0	**23**	0
0	**26**	0	**28**	0	**30**	0	**32**
33	0	**35**	0	**37**	0	**39**	0
0	**42**	0	**44**	0	**46**	0	**48**
49	0	**51**	0	**53**	0	**55**	0
0	**58**	0	**60**	0	**62**	0	**64**

Figure 4. Light-colored positions (bolded) on valid board b (See Figure 2 for all board position numbers).

0	**2**	0	**4**	0	**6**	7	**8**
9	0	**11**	0	**13**	0	**15**	0
0	**18**	0	**20**	0	**22**	0	**24**
25	0	**27**	0	**29**	0	**31**	0
0	**34**	0	**36**	0	**38**	0	**40**
41	0	**43**	0	**45**	0	**47**	0
0	**50**	0	**52**	0	**54**	0	**56**
57	0	**59**	0	**61**	0	**63**	0

Figure 5. Dark-colored positions (bolded) on valid board b (See Figure 2 for all board position numbers).

If b is valid, then the regular white pawns 2–9 or the regular black pawns 18–25 cannot be in the metamorphic positions 1–8 and 57–64 (See Figures 1 and 2). Let

$$\rho_\mu \quad = \quad \mathtt{ggn}(1,8,1) \otimes_r \mathtt{ggn}(57,64,1)$$

be the G-number of positions 1–8 and 57–64. The primitive recursive predicate

$$\begin{aligned} P_5(b) \equiv \quad & \{(\forall j \in \mathtt{ggn}(2,9,1))\{j \notin b \vee (\mathtt{psn}(j,b))_1 \notin \rho_\mu\} \wedge \\ & \{(\forall j \in \mathtt{ggn}(18,25,1))\{j \notin b \vee (\mathtt{psn}(j,b))_1 \notin \rho_\mu\}\} \end{aligned}$$

holds of b when the regular white and black pawns do not occupy the positions they cannot occupy either because they cannot move backward to reach those positions or because they undergo metamorphosis on those positions. In summary, the primitive recursive predicate

$$\mathtt{vld}(b) \equiv \mathtt{Lt}(b) = 64 \wedge P_1(b) \wedge P_2(b) \wedge P_3(b) \wedge P_4(b) \wedge P_5(b) \tag{43}$$

holds if b is a valid 64-cell chess board. Thus, if b_1 and b_2 are the boards in Figures 1 and 3, respectively, then $\mathtt{vld}(b_1) = \mathtt{vld}(b_2) = 1$.

4.4. Diagonals, Rows, Columns

A position $1 \leq k \leq 64$ on valid b is a member of at most two diagonals. Let $k' \in \{1,2\}$ and $d_{k,k'}$ be the diagonals (if $k' = 2$) or the diagonal that include k. For example, $k = 62$ is a member of two diagonals: $d_{62,1} = [48,55,62]$ and $d_{62,2} = [17,26,35,44,53,62]$ (See Figure 2); and $k = 1$ is a member of the diagonal $d_{1,1} = [1,10,19,28,37,46,55,64]$ (i.e., the first main diagonal).

Let

$$D = [\langle 1, [d_{1,1}]\rangle,\ \langle 2, [d_{2,1}, d_{2,2}]\rangle,\ \langle 3, [d_{3,1}, d_{3,2}], \rangle,\ \ldots,\ \langle 64, [d_{64,1}]\rangle]$$

be the G-number that pairs each position k with the diagonals that contain it.

Since each position is a member of exactly one row and exactly one column, we define G-numbers R and C to pack all rows and columns. For example, 7 is in the row $[1,2,3,4,5,6,7,8]$ and the column $[7,15,23,31,39,47,55,63]$.

Let

$$\begin{aligned} F_8(0) \quad & = \quad 1 \\ F_8(t+1) \quad & = \quad F_8(t) + 8 \end{aligned}$$

and

$$F_9(r) = \begin{cases} \mathtt{ggn}(F_8(r \dot{-} 1), F_8(r \dot{-} 1) + 7, 1) & \text{if } 1 \leq r \leq 8, \\ 0 & \text{otherwise.} \end{cases}$$

Since $F_8(\cdot)$ is primitive recursive by definition, $F_9(\cdot)$ is primitive recursive by the DCT. Thus,

$$\begin{aligned} F_9(1) &= \mathtt{ggn}(F_8(0), F_8(0)+7, 1) = \textstyle\prod_{j=1}^{8} \pi(j)^j; \\ F_9(2) &= \mathtt{ggn}(F_8(1), F_8(1)+7, 1) = \textstyle\prod_{j=9}^{16} \pi(j)^j; \\ F_9(8) &= \mathtt{ggn}(F_8(7), F_8(7)+7, 1) = \textstyle\prod_{j=57}^{64} \pi(j)^j. \end{aligned}$$

We place all row numbers on valid b into G-number R to pair each row number with the G-number of all positions in that row:

$$R = \textstyle\prod_{r=1}^{8} \langle r, F_9(r) \rangle.$$

For each $1 \leq i \leq 8$, we define eight primitive recursive functions (one for each column)

$$\begin{aligned} \kappa_i(0) &= i, \\ \kappa_i(t+1) &= \kappa_i(t) + 8 \end{aligned}$$

and let

$$c_i = \prod_{j=\kappa_i(0)}^{\kappa_i(7)} \pi(j)^j.$$

We place all column numbers into the G-number C that pairs each column number with the G-number of all positions in the column:

$$C = \textstyle\prod_{i=1}^{8} \langle i, c_i \rangle.$$

The primitive recursive functions in (44)–(46) return the G-number of the diagonals, rows, and columns, respectively, for a valid position k on b.

$$\mathtt{dgs}(k) = r((D)_{\mathtt{asc}(k,D)}) \tag{44}$$

$$\mathtt{rws}(k) = r((R)_{\mathtt{asc}(k,R)}) \tag{45}$$

$$\mathtt{cls}(k) = r((C)_{\mathtt{asc}(k,C)}) \tag{46}$$

Thus,

$$\begin{aligned} \mathtt{dgs}(62) &= [d_{62,1}, d_{62,2}] \\ &= [[48, 55, 62], [17, 26, 35, 44, 53, 62]]; \\ (\mathtt{dgs}(62))_1 &= d_{62,1}; \\ (\mathtt{dgs}(62))_2 &= d_{62,2}; \\ \mathtt{rws}(62) &= [57, 58, 59, 60, 61, 62, 63, 64]; \\ \mathtt{cls}(62) &= [6, 14, 22, 30, 38, 46, 54, 62]. \end{aligned}$$

Given positions k_1 and k_2, let

$$z_d = \mathtt{dgs}(k_1) \otimes_r \mathtt{dgs}(k_2)$$

and

$$F_{10}(k_1, k_2) = \min_{t \leq Lt(z_d)} \{k_1 \in (z_d)_t \wedge k_2 \in (z_d)_t\}.$$

The primitive recursive function $\mathtt{dg}(k_1, k_2)$ returns diagonal d that contains both k_1 and k_2 if d exists.

$$\mathtt{dg}(k_1, k_2) = (z_d)_{F_{10}(k_1,k_2)} \tag{47}$$

Let $z_r = [\mathtt{rws}(k_1)] \otimes_r [\mathtt{rws}(k_2)]$ and

$$F_{11}(k_1, k_2) = \min_{t \leq Lt(z_r)} \{k_1 \in (z_r)_t \wedge k_2 \in (z_r)_t\}.$$

The primitive recursive function $\mathtt{rw}(k_1, k_2)$ returns row r that contains both k_1 and k_2 if r exists.

$$\mathtt{rw}(k_1, k_2) = (z_r)_{F_{11}(k_1,k_2)} \tag{48}$$

Let $z_c = [\mathtt{cls}(k_1)] \otimes_r [\mathtt{cls}(k_2)]$ and

$$F_{12}(k_1, k_2) = \min_{t \le Lt(z_c)} \{k_1 \in (z_c)_t \wedge k_2 \in (z_c)_t\}.$$

The primitive recursive function $\mathtt{cl}(k_1, k_2)$ returns column c that contains both k_1 and k_2 if c exists.

$$\mathtt{cl}(k_1, k_2) = (z_c)_{F_{12}(k_1,k_2)} \tag{49}$$

4.5. Potentially Reachable Positions

The primitive recursive predicate

$$\mathtt{oc}(x, y) \equiv \{\mathtt{wp}(x) \wedge \mathtt{bp}(y)\} \vee \{\mathtt{wp}(y) \wedge \mathtt{bp}(x)\} \tag{50}$$

holds for valid pieces x and y of the opposite colors.

Let j, k, and b denote a valid piece, a valid position, and a valid board, respectively. Position $k' \neq k$ on b is *potentially reachable* for j from k if j can move from k to k' on b. Position k' can be empty or occupied by a different piece.

Let L_j^k be the G-number of the positions potentially reachable by j from k on b. For example, L_{15}^1 is the G-number of the positions potentially reachable by bishop 15 from position 1. The G-numbers $L_{15}^k, k \in [10, 19, 28, 37, 46, 55, 64]$, denote the positions potentially reachable by 15 from the other positions of the first main diagonal. We note that L_j^k is defined regardless of any b in the sense that whether k' is potentially reachable for j from k depends only on the move rules for j and not on the positions of the other pieces on b.

Thus, for bishop 15 positioned anywhere on the first main diagonal, we have

$$\begin{aligned}
L_{15}^1 &= \langle 1, [10, 19, 28, 37, 46, 55, 64]\rangle;\\
L_{15}^{10} &= \langle 10, [1, 3, 17, 19, 28, 37, 46, 55, 64]\rangle;\\
L_{15}^{19} &= \langle 19, [1, 10, 5, 12, 26, 33, 28, 37, 46, 55, 64]\rangle;\\
L_{15}^{28} &= \langle 28, [1, 10, 19, 7, 14, 21, 35, 42, 49, 37, 46, 55, 64]\rangle;\\
L_{15}^{37} &= \langle 37, [1, 10, 19, 28, 16, 23, 30, 44, 51, 58, 46, 55, 64]\rangle;\\
L_{15}^{46} &= \langle 46, [1, 10, 19, 28, 37, 32, 39, 53, 60, 55, 64]\rangle;\\
L_{15}^{55} &= \langle 55, [1, 10, 19, 28, 37, 46, 48, 62, 64]\rangle;\\
L_{15}^{64} &= \langle 64, [1, 10, 19, 28, 37, 46, 55]\rangle.
\end{aligned}$$

We recall that ζ_{lps} in (41) is the G-number of the light-colored positions on b. Since 15 is light-colored, the G-number

$$L_{15} = \otimes_r|_{k \in \zeta_{lps}} \left[L_{15}^k\right]$$

contains all positions potentially reachable by 15. We observe that if j is another real or metamorphic light-colored black bishop (e.g., $j = 28$), then

$$L_j = L_{15}.$$

For the dark-colored real bishops 12 and 31 (or any dark-colored metamorphic bishop j), we can analogously define $L_{12} = L_{31} = L_j$ using ζ_d in (42) (i.e., the G-number of the dark-colored positions on valid b).

In general, for each valid piece j we define the G-number of all potentially reachable positions on b and place all such G-numbers into L in (51) of the pairs $\langle j, L_j\rangle$, where L_j is the G-number of all reachable positions for j on b.

$$L = \left(\otimes_r \Big|_{j=2}^{33} [\langle j, L_j \rangle] \right) \otimes_r \left(\otimes_r \Big|_{j \in \zeta_\mu^w \otimes_r \zeta_\mu^b} [\langle j, L_j \rangle] \right) \tag{51}$$

The function

$$\mathtt{rp}(j,k,b) = \begin{cases} r(\mathtt{asc}(k, r(\mathtt{asc}(j,L)))) & \text{if } \mathtt{vlp}(j) \wedge \mathtt{vld}(b) \\ 0 & \text{otherwise} \end{cases} \tag{52}$$

returns the G-number of all potentially reachable positions for j from k on b. The function $\mathtt{rp}(\cdot)$ is primitive recursive by the DCT.

Thus, if b and b' are boards in Figures 1 and 3, respectively, then (See Figures 6 and 7)

$$\begin{aligned} \mathtt{rp}(16, 63, b) &= [53, 46, 48]; \\ \mathtt{rp}(16, 46, b') &= [61, 52, 36, 29, 31, 40, 56, 63]. \end{aligned}$$

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	**46**	47	**48**
49	50	51	52	**53**	54	55	56
57	58	59	60	61	62	*63*	64

Figure 6. $rp(16, 63, b) = [53, 46, 48]$, where b is the board in Figure 1.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	**29**	30	**31**	32
33	34	35	**36**	37	38	39	**40**
41	42	43	44	45	*46*	47	48
49	50	51	**52**	53	54	55	**56**
57	58	59	60	**61**	62	**63**	64

Figure 7. $rp(16, 46, b') = [61, 52, 36, 29, 31, 40, 56, 63]$; b' is in Figure 3.

4.6. Actual Reachability for Bishop 15

If $k' \in \mathtt{rp}(j,k,b)$, it may be occupied by $j' \neq j$ of the same color. For example, although $52 \in \mathtt{rp}(16, 46, b')$, $(b')_{52} = 5 \neq 1$, i.e., it is occupied by pawn 5 of the same color as knight 16 and, consequently, 16 cannot move from 46 to 52 on b'. To put it differently, 52 is not *actually* reachable for 16 from 46 on b'.

The characteristics of *potential reachability* and *actual reachability*, while conceptually related by the rules of chess, are different in that the potential reachability of k' for j from k holds on any valid b whereas the actual reachability of k' for j from k depends not only on whether $k' \in \mathtt{rp}(j,k,b)$ but also on $(b)_{k'}$.

As an extended example, let us consider the characteristic of actual reachability for bishop 15 to determine if this characteristic is pr-computable. For any valid position k of 15 on b, the positions in the G-number of actually reachable positions for 15 from k depend on the light-colored bishop move rules and require us to take into account the positions of the other pieces of the same color on b.

Let the primitive recursive function

$$\mathtt{cpsn}(j,b) = \begin{cases} (\mathtt{psn}(j,b))_1 & \text{if } \mathtt{vlp}(j) \wedge \mathtt{vld}(b), \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

return the current position of valid piece j on valid board b so that if $j \in b$, then

$$(b)_{\mathtt{cpsn}(j,b)} = j.$$

Let $\mathtt{cpsn}(15, b) = k_{15}$. We seek to compute the G-number of actually reachable positions for 15 from k_{15} on b. Let $k \neq k_{15}$ be a valid position. Then $\mathtt{dg}(k_{15}, k)$ in (49) returns diagonal d that contains both k_{15} and k. If d does not exist (i.e., $d = 0$), then k is not actually reachable for 15 from k_{15} on b. If d exists, we need to ensure that either k is empty on b (i.e., $(b)_k = 1$) or is occupied by a piece of the opposite color $\mathtt{oc}((b)_k, 15) = 1$ and 15 can capture $(b)_k$. Furthermore, if d exists, we need to ensure that there is no piece either of the same or opposite color between k_{15} and k on d. The primitive recursive predicate

$$\begin{array}{rcl} P_6(k, k_{15}, b) & \equiv & \mathtt{vld}(b) \\ & \wedge & k \neq k_{15} \\ & \wedge & k \in \mathtt{rp}(15, k_{15}, b) \\ & \wedge & \mathtt{dg}(k_{15}, k) \neq 0 \\ & \wedge & (\forall k' \in \mathtt{dg}(k_{15}, k))\{ \\ & & \{k' = k \vee k' = k_{15}\} \vee (b)_{k'} = 1\} \\ & \wedge & \{(b)_k = 1 \vee \mathtt{oc}(15, (b)_k)\} \end{array}$$

holds on b if k is potentially reachable for 15 from k_{15} and is on the same diagonal with k_{15} and every position k' between k and k_{15} on the diagonal is empty and k itself is either empty or occupied by a black piece. The primitive recursive predicate

$$\mathtt{arp}_{15}(k, k_{15}, b) \equiv P_6(k, k_{15}, b)$$

holds when k is actually reachable for 15 from k_{15} on valid b.

Let b' be the board in Figure 3. Then

$$\mathtt{arp}_{15}(48, 62, b') = 0,$$

because $\mathtt{dg}(62, 48) = d_{62,1} = [48, 55, 62]$, but pawn 8 in position 55 is of the same color with 15, which makes $P_6(48, 62, b')$ false. On the other hand,

$$\mathtt{arp}_{15}(k, 62, b') = 1,$$

for any $k \in \{17, 26, 35, 44, 53\}$. Let b be a board and let

$$\mathtt{arp}'_{15}(k) \equiv \mathtt{arp}_{15}(k, k_{15}, b).$$

Then for any k_{15} on b,

$$\mathtt{map}_{\mathtt{arp}'_{15}(k)}(\zeta_{lps})$$

computes in a primitive recursive fashion the G-number of all positions actually reachable for 15 from k_{15} on b (See Equation (41) for ζ_{lps}), which furnishes us the following lemma.

Lemma 1. *Actual reachability for bishop 15 is pr-decidable and pr-computable.*

4.7. Actual Reachability

Let j be a valid piece on valid b. Let $j = (b)_{k_j}$, where

$$k_j = \mathtt{cpsn}(j, b) \tag{54}$$

is the pr-computable position of j on b. Thus, k_{30} is the position of the black king on b and k_{14} is the position of the white king on b.

Let $k \in \mathtt{rp}(j, k_j, b)$. By the rules of chess, j can actually reach finitely many positions from k_j. Consequently, whether k is actually reachable for j from k_j on b can be decided in a primitive recursive fashion with the functions and predicates in Sections 2 and 3 and

$\mathtt{oc}(\cdot)$, $\mathtt{vld}(\cdot)$, $\mathtt{dg}(\cdot)$, $\mathtt{rw}(\cdot)$, $\mathtt{cl}(\cdot)$, $\mathtt{psn}(\cdot)$, $\mathtt{rp}(\dot{)}$ through appropriate Boolean combinations of primitive recursive predicates and applications of the DCT in the way similar to the one constructed for bishop 15 in the previous section. Hence, we have the following lemma.

Lemma 2. *Let* $\mathtt{vlp}(j) = \mathtt{vld}(b) = 1$ *and* $k \in \mathtt{rp}(j, k_j, b)$. *Then* $\mathtt{arp}_j(k, k_j, b)$ *is primitive recursive.*

Since there are finitely many regular and metamorphic pieces, there are finitely many predicates $\mathtt{arp}_j(\cdot)$. Let $z = \mathtt{rp}(j, k_j, b)$. The function

$$\begin{array}{rcl} \mathtt{arx}(z, j, k_j, b, 0) & = & [\,], \\ \mathtt{arx}(z, j, k_j, b, t+1) & = & F_{13}(z, j, k_j, b, \\ & & \quad \mathtt{arx}(z, j, k_j, b, t), \\ & & \quad s(t)), \end{array}$$

where

$$F_{13}(z, j, k_j, b, y, i) = \begin{cases} y & \text{if } (z)_i \notin \mathtt{arp}_j(j, k_j, b), \\ [(z)_i] \otimes_l y & \text{if } (z)_i \in \mathtt{arp}_j(j, k_j, b) \end{cases}$$

returns the G-number of all positions in z that are actually reachable for j from k_j on b. The function $F_{13}(\cdot)$ is primitive recursive by Lemma 1 and the DCT, which makes $\mathtt{arx}(\cdot)$ also primitive recursive.

The function $\mathtt{arps}(\cdot)$ in (55) returns the G-number of actually reachable positions for j from k_j on b. This function is primitive recursive, because it is composed from primitive recursive functions.

$$\mathtt{arps}(j, k_j, b) = \mathtt{arx}(\mathtt{rp}(j, k_j, b), j, k_j, b, \mathtt{Lt}(\mathtt{rp}(j, k_j, b))) \tag{55}$$

Thus, if b and b' are the boards in Figures 1 and 3, respectively, then

$$\begin{array}{l} \mathtt{arps}(15, 62, b) = [\,]; \\ \mathtt{arps}(16, 63, b) = [46, 48]; \\ \mathtt{arps}(15, 62, b') = [17, 26, 35, 44, 53]; \\ \mathtt{arps}(16, 46, b') = [29, 31, 36, 40, 63]. \end{array}$$

We have the following lemma.

Lemma 3. *Actual reachability for valid piece j on valid board b is pr-decidable and pr-computable.*

4.8. Checks, Mates, and Stalemates

Black king 30 on valid b is checked when there is a white piece $j \neq 14$ (i.e., different from white king 14), for which k_{30} (See Equation (54)) is actually reachable. The primitive recursive predicate

$$\mathtt{bkc}(b) \equiv (\exists j \in \zeta_{wp})\{j \neq 14 \wedge k_{30} \in \mathtt{arps}(j, k_j, b)\} \tag{56}$$

holds of b when king 30 is checked on it (See Equation (35) for ζ_{wp}). King 30 is mated if, when checked, it cannot move to any position that is not actually reachable by some white piece j from the latter's current position. The primitive recursive predicate

$$\mathtt{bkm}(b) \equiv \mathtt{bkc}(b) \wedge (\forall k \in \zeta_{ps})\{k \notin \mathtt{arps}(30, k, b) \vee (\exists j \in \zeta_{wp})\{k \in \mathtt{arps}(j, k_j, b)\}\} \tag{57}$$

holds of b when king 30 is mated on it (See Equation (32) for ζ_{ps}). The primitive recursive predicate

$$\mathtt{wkc}(b) \equiv (\exists j \in \zeta_{bp})\{j \neq 30 \wedge k_{14} \in \mathtt{arps}(j, k_j, b)\} \tag{58}$$

holds of b when white king 14 is checked on it (See Equation (35) for ζ_{wp}); and the primitive recursive predicate

$$\texttt{wkm}(b) \equiv \texttt{wkc}(b) \wedge (\forall k \in \zeta_{ps})\{k \notin \texttt{arps}(14, k, b) \vee (\exists j \in \zeta_{bp})\{k \in \texttt{arps}(j, k_j, b)\}\} \quad (59)$$

holds of b when 14 is mated on it.

A stalemate occurs when the player whose turn it is to make a move on b is not in check, yet has no legal move to make. In other words, all the player's pieces have no actually reachable positions. The primitive recursive predicate

$$\texttt{bsm}(b) \equiv \neg\texttt{bck}(b) \wedge (\forall j \in \zeta_{bp})\{\texttt{Lt}(\texttt{arps}(j, k_j, b)) = 0\} \quad (60)$$

holds if b is the black stalemate; and the primitive recursive predicate

$$\texttt{wsm}(b) \equiv \neg\texttt{wck}(b) \wedge (\forall j \in \zeta_{wp})\{\texttt{Lt}(\texttt{arps}(j, k_j, b)) = 0\} \quad (61)$$

holds if b is the white stalemate.

4.9. Dead Position

The dead position rule (DPR) holds for valid b when neither player can checkmate the opponent by any series of moves. The DPR can be broken into four cases: (**1**) b contains only 14 (white king) and 30 (black king); (**2**) b contains only 14, 30 and a bishop; (**3**) b contains only 14, 30 and a knight; (**4**) b contains only 14, 30, one white bishop, and one black bishop such that one bishop is light-colored and the other one-dark-colored.

The primitive recursive predicate

$$\texttt{dp}_1(b) \equiv \texttt{vld}(b) \wedge (\forall j \in \zeta_{cp})\{j = 14 \vee j = 30 \vee j \notin b\} \quad (62)$$

holds for case 1.

We break case 2 into five sub-cases: (**2a**) b contains only 14, 30, 12; (**2b**) b contains only 14, 30, 15; (**2c**) b contains only 14, 30, 28; (**2d**) b contains only 14, 30, 31; (**2e**) b contains only 14, 30, and exactly one metamorphic bishop.

The primitive recursive predicate

$$\begin{aligned} \texttt{dpx}(b, j) \equiv \quad & \{\texttt{vld}(b) \wedge j \in b \wedge j \neq 14 \wedge j \neq 30\} \wedge \\ & (\forall i \in \zeta_{cp})\{i = 14 \vee i = 30 \vee i = j \vee i \notin b\}\} \end{aligned} \quad (63)$$

holds when b contains 14, 30, and exactly one other piece j. The primitive recursive predicate

$$\begin{aligned} \texttt{dp}_2(b) \equiv \ & \texttt{dpx}(b, 12) \ \vee \texttt{dpx}(b, 15) \ \vee \\ & \texttt{dpx}(b, 28) \ \vee \ \texttt{dpx}(b, 31) \ \vee \\ & (\exists j \in \zeta_{cp})\{\texttt{bh}_\mu(j) \wedge \texttt{dpx}(b, j)\} \end{aligned} \quad (64)$$

holds for case 2 (See Equation (40) for $\texttt{bh}_\mu(\cdot)$).

We break case 3 of the DPR into five sub-cases: (**3**) b contains only 14, 30, 11; (**3b**) b contains only 14, 30, 16; (**3c**) b contains only 14, 30, 27; (**3d**) b contains only 14, 30, 32. (**3e**) b contains only 14, 30, and exactly one metamorphic knight.

The primitive recursive predicate

$$\begin{aligned} \texttt{dp}_3(b) \equiv \ & \texttt{dpx}(b, 11) \ \vee \texttt{dpx}(b, 16) \ \vee \\ & \texttt{dpx}(b, 27) \ \vee \texttt{dpx}(b, 32) \ \vee \\ & (\exists j \in \zeta_{cp})\{\texttt{kn}_\mu(j) \wedge \texttt{dpx}(b, j)\} \end{aligned} \quad (65)$$

holds for case 3.

We break case 4 of the DPR into three sub-cases: (**4a**) b contains 14, 30, 15, 31; (**4b**) b contains 14, 30, 12, 28; (**4c**) b contains 14, 30, and two metamorphic bishops one of which is light-colored and the other dark-colored. Let the primitive recursive predicate

$$\begin{array}{l}\texttt{dpxx}(b,j',j'') \equiv \{\texttt{vld}(b) \ \wedge \\ j' \in b \ \wedge j'' \in b \ \wedge \ j' \neq j'' \ \wedge \\ j' \neq 14 \ \wedge j' \neq 30 \ \wedge \\ j'' \neq 14 \ \wedge j'' \neq 30 \ \wedge \\ (\forall i \in \zeta_{cp})\{\{i = 14 \ \vee \ i = 30 \vee i = j' \vee i = j''\} \ \vee \ i \notin b\}\}\end{array} \tag{66}$$

hold of two different pieces j' and j'' on valid b that are the only two pieces on b different from 14 and 30. The primitive recursive predicate

$$\texttt{dp}_4(b) \equiv \texttt{dpxx}(b,15,31) \vee \texttt{dpxx}(b,12,28) \vee (\exists j' \in \zeta_\mu^{lcb})(\exists j'' \in \zeta_\mu^{dcb})\{\texttt{dpxx}(b,j',j'')\} \tag{67}$$

holds for case 4. The primitive recursive predicate

$$\texttt{dp}(b) = \texttt{dp}_1(b) \vee \texttt{dp}_2(b) \vee \texttt{dp}_3(b) \vee \texttt{dp}_4(b) \tag{68}$$

holds if b is a valid dead position board.

There are two more types of draw in chess: repetition and the 50-move rule. A draw by repetition is achieved when the same position occurs three times in a row with the same player to move. The 50-move rule states that a game is a draw when the last 50 moves contain no capture or pawn move. We will consider these types of draw after we formalize the concept of game history.

4.10. Moves

Let the primitive recursive predicate

$$\begin{array}{l}P_7'(j,k,b) \equiv \texttt{vld}(b) \wedge j \in b \ \wedge \ k \in \texttt{arps}(j,k_j,b) \wedge \neg\texttt{wkm}(b) \wedge \neg\texttt{bkm}(b) \ \wedge \\ \qquad\qquad \neg\texttt{wsm}(b) \wedge \neg\texttt{bsm}(b) \wedge \neg\texttt{dp}(b)\end{array}$$

hold when position k is actually reachable for piece j on valid b from k_j (as defined in (54)) and b is not a mate, a stalemate, or a draw. Let

$$\boxed{b}_k^j = \texttt{set}(\texttt{set}(b,k,j),k_j,1). \tag{69}$$

The above equation defines a primitive recursive function that calculates the board obtained from b after j is placed in position k and position k_j is set to 1. Thus, if b_0 is the starting board in Figure 1, then

$$\boxed{b}_{37}^6 = \texttt{set}(\texttt{set}(b_0,37,6),53,1)$$

is the board after the white player moves pawn 6 from position 53 to position 37.

Let the primitive recursive predicate

$$P_7''(j,k,b) \equiv \neg\left\{j \in \zeta_{wp} \wedge \texttt{wkc}\left(\boxed{b}_k^j\right)\right\} \ \wedge \neg\left\{j \in \zeta_{bp} \wedge \texttt{bkc}\left(\boxed{b}_k^j\right)\right\}$$

hold if the move of j into k from k_j does not result a board where the king of the same color with j is checked, because such moves are not allowed by the rules of chess.

Let

$$P_7(j,k,b) \equiv P_7'(j,k,b) \wedge P_7''(j,k,b)$$

and

$$\left\{b\right\}_k^j = \begin{cases}\boxed{b}_k^j & \text{if } P_7(j,k,b) \\ 0 & \text{otherwise.}\end{cases} \tag{70}$$

The function defined in (70) is primitive recursive by the DCT. Thus, if b_0 is the board in Figure 1, then the board in Figure 3 is

$$\left\{\left\{\left\{\left\{b_0\right\}_{37}^{6}\right\}_{29}^{22}\right\}_{46}^{16}\right\}_{22}^{32} = \boxed{\boxed{\boxed{b_0}_{37}^{6}}_{29}^{22}}_{46}^{16}{}_{22}^{32}.$$

The right-hand side of the above equality denotes the board obtained from b_0 with four moves: (1) white pawn 6 to position 37; (2) black pawn 22 to position 29; (3) white knight 16 to position 46; and (4) black knight 32 to position 22; or (1) e4 e5; (2) Kf3 Kf6 in standard chess notation.

We will denote the white player by 1 and the black player by 2. If x is a player, then $\bar{x}$ denotes the player's opponent. Thus, if $x = 1$, then $\bar{x} = 2$; if $x = 2$, then $\bar{x} = 1$. Let

$$\left\langle\left\langle\left\langle b, \left\{b\right\}_k^j\right\rangle, x\right\rangle, \left\langle j, \left\langle k_j, k\right\rangle\right\rangle\right\rangle \tag{71}$$

denote the move of piece j by player x from k_j to k on b that results in $\{b\}_k^j$. Thus,

$$\left\langle\left\langle\left\langle b_0, \left\{b_0\right\}_{37}^{6}\right\rangle, 1\right\rangle, \left\langle 6, \left\langle 53, 37\right\rangle\right\rangle\right\rangle$$

is the move of pawn 6 from 53 to 37 by player 1 on b_0 that results in

$$\left\{b_0\right\}_{37}^{6} = \boxed{b_0}_{37}^{6}.$$

The primitive recursive function $\texttt{mvp}(\cdot)$ in (72) constructs the required move number for valid b, x, j, k.

$$\texttt{mvp}(b, x, j, k) = \left\langle\left\langle\left\langle b, \left\{b\right\}_k^j\right\rangle, x\right\rangle, \left\langle j, \left\langle k_j, k\right\rangle\right\rangle\right\rangle \tag{72}$$

Let

$$z = \texttt{mvp}(b, x, j, k)$$

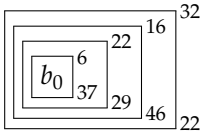
be a move constructed from valid b, x, j, and k. Then the primitive recursive functions $\texttt{pcb}(\cdot)$, $\texttt{scb}(\cdot)$, $\texttt{plr}(\cdot)$, and $\texttt{mvi}(\cdot)$ in (73) return the predecessor board, the successor board, the player, the piece and the piece's positions of move z, respectively.

$$\begin{aligned} \texttt{pcb}(z) &= l(l(l(z))) \\ \texttt{scb}(z) &= r(l(l(z))) \\ \texttt{plr}(z) &= r(l(z)) \\ \texttt{mvi}(z) &= r(z) \end{aligned} \tag{73}$$

Thus, if b_0 be the board in Figure 1, the G-number

$$\begin{array}{l} [\\ \texttt{mvp}(b_0, 1, 6, 37), \texttt{mvp}(\texttt{scb}(\texttt{mvp}(b_0, 1, 6, 37)), 2, 22, 29), \\ \texttt{mvp}(\texttt{scb}(\texttt{mvp}(\texttt{scb}(\texttt{mvp}(b_0, 1, 6, 37)), 2, 22, 29)), 1, 16, 46), \\ \texttt{mvp}(\texttt{scb}(\texttt{mvp}(\texttt{scb}(\texttt{mvp}(\texttt{scb}(\texttt{mvp}(b_0, 1, 6, 37)), 2, 22, 29)), 1, 16, 46)), 2, 32, 22) \\ ] \end{array}$$

contains a sequence of moves from b_0 in Figure 1 to the board



in Figure 3.

Let M_b^x be the G-number of all possible moves that player x can do on b if it is x's turn to play. Equations (74) and show us how M_b^1 and M_b^2 are pr-computable.

$$\begin{aligned} M_b^1 &= \otimes_r\Big|_{j\in\zeta_{wp}} \otimes_r \Big|_{k\in\mathtt{arps}(j,k_j,b)} [\mathtt{mvp}(b,1,j,k)] \\ M_b^2 &= \otimes_r\Big|_{j\in\zeta_{bp}} \otimes_r \Big|_{k\in\mathtt{arps}(j,k_j,b)} [\mathtt{mvp}(b,2,j,k)] \end{aligned} \tag{74}$$

Let the primitive recursive predicate

$$P_8(b) \equiv \mathtt{vld}(b) \wedge \neg\mathtt{wkm}(b) \wedge \neg\mathtt{bkm}(b) \wedge \neg\mathtt{wsm}(b) \wedge \neg\mathtt{bsm}(b) \wedge \neg\mathtt{dp}(b)$$

hold when b is not a mate, a stalemate, or a dead position. The function $\mathtt{mvs}(\cdot)$ in (75), which is primitive recursive by the DCT, maps board b to the G-number of all possible moves obtained from it by exactly one move of player x.

$$\mathtt{mvs}(x,b) = \begin{cases} \mathtt{rm}(0,M_b^1) & \text{if } x = 1 \wedge P_8(b), \\ \mathtt{rm}(0,M_b^2) & \text{if } x = 2 \wedge P_8(b), \\ 0 & \text{otherwise.} \end{cases} \tag{75}$$

4.11. History

An *epoch* of a D2PBG is the G-number of all possible valid boards that can be obtained from each board in the previous epoch with exactly one move by the player whose turn it is to play on the boards of the previous epoch. The *history* of a D2PBG is the G-number of finitely many epochs.

Let b_0 be the initial board in Figure 1. Let the history of chess start at epoch

$$E_0 = \langle EM_0, 1\rangle, \tag{76}$$

where

$$EM_0 = [\langle\langle 0, b_0\rangle, 0\rangle, \langle 0, \langle 0, 0\rangle\rangle].$$

We observe that

$$\mathtt{scb}((l(E_0))_1) = \mathtt{scb}((EM_0)_1) = b_0$$

and $r(E_0) = 1$ denotes player 1 whose turn it is to play on $\mathtt{scb}((l(E_0))_1)$.

Epoch E_t, $t > 0$, is

$$E_t = \langle EM_t, x\rangle, \tag{77}$$

where EM_t is pr-computable from E_{t-1} as

$$EM_t = \otimes_r|_{m\in l(E_{t-1})}\ \mathtt{mvs}(\mathtt{r}(E_{t-1}), \mathtt{scb}(m)). \tag{78}$$

Thus, epoch E_1, is pr-computable from E_0 as

$$E_1 = \langle EM_1, 2\rangle,$$

where

$$EM_1 = \otimes_r|_{m\in l(E_0)}\ \mathtt{mvs}(1, \mathtt{scb}(m)).$$

Epoch E_2 is pr-computable from E_1 as

$$E_2 = \langle EM_2, 1\rangle,$$

where

$$EM_2 = \otimes_r|_{m\in l(E_1)}\ \mathtt{mvs}(2, \mathtt{scb}(m)).$$

To generalize, we let

$$F_{14}(E_t, t) = \begin{cases} \langle EM_{s(t)}, 1\rangle & \text{if } 2|s(t), \\ \langle EM_{s(t)}, 2\rangle & \text{if } 2\nmid s(t), \end{cases}$$

and define epoch E_t as a primitive recursive function in (79).

$$\begin{array}{rcl} E_0 & = & \langle EM_0, 1\rangle, \\ E_{t+1} & = & F_{14}(E_t, t). \end{array} \tag{79}$$

Thus,

$$\begin{array}{l} E_0 = \langle EM_0, 1\rangle; \\ E_1 = F_{14}(E_0, 0) = \langle EM_1, 2\rangle; \\ E_2 = F_{14}(E_1, 1) = \langle EM_2, 1\rangle; \\ E_3 = F_{14}(E_2, 2) = \langle EM_3, 2\rangle; \\ E_4 = F_{14}(E_3, 3) = \langle EM_4, 1\rangle; \\ E_5 = F_{14}(E_4, 4) = \langle EM_5, 2\rangle; \\ \ldots \end{array}$$

Given an epoch number t, the history of chess, $\mathbb{H}_t$, is pr-computable as

$$\mathbb{H}_t = \otimes_r|_{k=0}^{t}[E_i] = [E_0, E_1, \ldots, E_{t-1}, E_t]. \tag{80}$$

Thus, for $t > 2$.

$$\begin{array}{rcl} \mathbb{H}_0 & = & [E_0]; \\ \mathbb{H}_1 & = & [E_0, E_1]; \\ \mathbb{H}_2 & = & [E_0, E_1, E_2]; \\ & \ldots & \\ \mathbb{H}_t & = & [E_0, E_1, E_2, \ldots, E_t]. \end{array}$$

4.12. Games

We define the primitive recursive predicate

$$\mathtt{eb}(b, t) \equiv (\exists m \in M_t)\{\mathtt{scb}(m) = b\} \tag{81}$$

to hold when b is a successor of some move in M_t, $t \geq 0$. We will refer to b as an epoch t board. Let the primitive recursive predicate

$$\begin{array}{l} \mathtt{scr}(b, b', t) \equiv \{t > 0 \wedge \mathtt{eb}(b, t-1) \wedge \mathtt{eb}(b', t) \wedge \\ \qquad (\exists m \in M_t)\{\mathtt{pcb}(m) = b\ \wedge\ \mathtt{scb}(m) = b'\}\} \end{array} \tag{82}$$

hold between boards b and b' if for some move $m \in M_t$ such that b is the move's predecessor and b' is its successor. In other words, epoch t board b' can be obtained from epoch $t-1$ board b in exactly one move by the player whose turn it is to play on b.

Let $z = [b_{i_1}, \ldots, b_{i_k}]$, $k \geq 2$, be a sequence of boards and let b_0 be the starting board. The primitive recursive predicate

$$\mathtt{hgm}(z) \equiv \mathtt{Lt}(z) \geq 2 \wedge (z)_1 = b_0 \wedge (\forall t \in \mathtt{ggn}(2, \mathtt{Lt}(z), 1))\{\mathtt{scr}((z)_{t-1}, (z)_t, t)\} \tag{83}$$

holds of z when it has at least two boards, its first board is b_0, and the predicate $\mathtt{scr}(\cdot)$ in (82) holds of every two consecutive boards in z. If $\mathtt{hgm}(z) = 1$ for some sequence of boards z, we refer to z as *chess game* G_z or simply as *game* G_z.

Consider the board sequences in Figures 8–11 and let

$$\begin{array}{l} z_1 = [b_0] \otimes_r z_1' = [b_0, b_1^1, b_2^1, b_3^1, b_4^1]; \\ z_2 = [b_0] \otimes_r z_2' = [b_0, b_1^2, b_2^2, b_3^2, b_4^2]; \\ z_3 = [b_0] \otimes_r z_3' = [b_0, b_1^3, b_2^3, b_3^3, b_4^3]; \\ z_3 = [b_0] \otimes_r z_4' = [b_0, b_1^4, b_2^4, b_3^4, b_4^4]. \end{array}$$

Then, since

$$\mathtt{hgm}(z_1) = \mathtt{hgm}(z_2) = \mathtt{hgm}(z_3) = \mathtt{hgm}(z_4) = 1,$$

G_{z_1}, G_{z_2}, G_{z_3}, and G_{z_4} are games.

Let z be a G-number of boards. The primitive recursive function

$$\mathtt{tlb}(z) = (z)_{\mathtt{Lt}(z)} \tag{84}$$

computes the last board of z. If G_z is a game, we will refer to the last board of G_z (i.e., $\mathtt{tlb}(G_z)$) as the *tail board* or the *tail* of G_z.

The primitive recursive function $J(\cdot)$ computes the G-number of all games up to epoch $t > 0$.

$$\begin{array}{lcl} J(0) & = & [[b_0]], \\ J(t+1) & = & F_{15}(J(t), t), \end{array} \tag{85}$$

where

$$F_{15}(Z, t) = \otimes_r|_{z \in Z} F_{16}(z, EM_{t+1}), \tag{86}$$

where

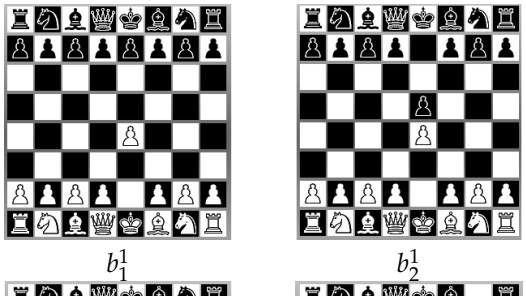
$$\begin{array}{rcl} F_{16}(z, EM) & = & F_{17}(z, \mathtt{map}_{\mathtt{pcb}(m)=\mathtt{tlb}(z)}(EM), \\ & & \quad \mathtt{Lt}(\mathtt{map}_{\mathtt{pcb}(m)=\mathtt{tlb}(z)}(EM))), \end{array} \tag{87}$$

where

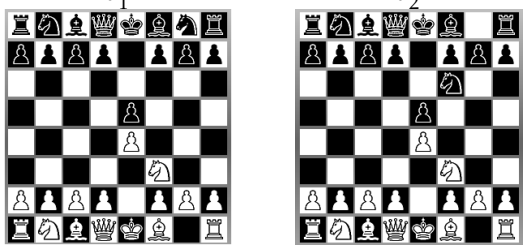
$$\begin{array}{lcl} F_{17}(z, EM, 0) & = & [\,], \\ F_{17}(z, EM, t+1) & = & F_{18}(z, EM, F_{17}(z, EM, t), t), \end{array} \tag{88}$$

where

$$F_{18}(z, EM, Z, t) = Z \otimes_r [z \otimes_r [\mathtt{scb}((EM)_{s(t)})]]. \tag{89}$$



b_1^1 b_2^1



b_3^1 b_4^1

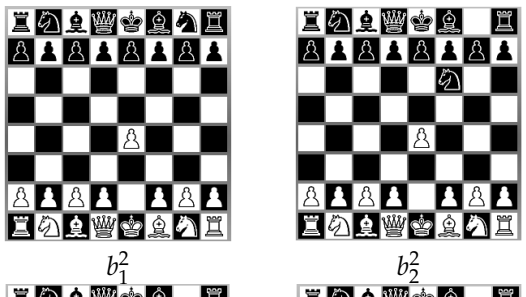
$$b_1^1 = \left\{ b_0 \right\}_{37}^{6} = \boxed{b_0}_{37}^{6}$$

$$b_2^1 = \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{29}^{22} = \boxed{\boxed{b_0}_{37}^{6}}_{29}^{22}$$

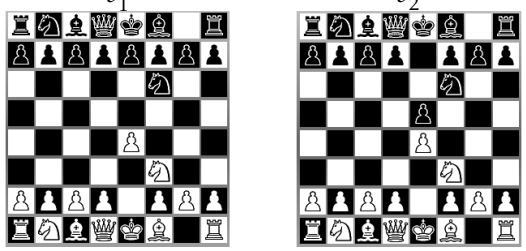
$$b_3^1 = \left\{ \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{29}^{22} \right\}_{46}^{16} = \boxed{\boxed{\boxed{b_0}_{37}^{6}}_{29}^{22}}_{46}^{16}$$

$$b_4^1 = \left\{ \left\{ \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{29}^{22} \right\}_{46}^{16} \right\}_{22}^{32} = \boxed{\boxed{\boxed{\boxed{b_0}_{37}^{6}}_{29}^{22}}_{46}^{16}}_{22}^{32}$$

Figure 8. Board sequence z_1' corresponding to the moves (1) e4 e5; (2) Kf3 Kf6.



b_1^2 b_2^2



b_3^2 b_4^2

Figure 9. *Cont.*

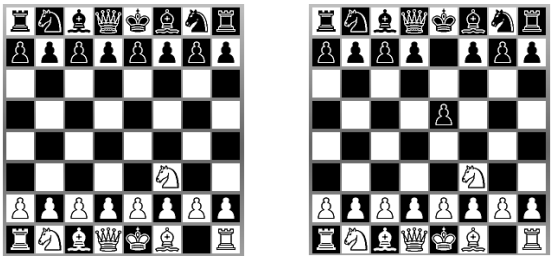
$$b_1^2 = \left\{ b_0 \right\}_{37}^{6} \quad = \quad \boxed{b_0}_{37}^{6}$$

$$b_2^2 = \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{22}^{32} \quad = \quad \boxed{\boxed{b_0}_{37}^{6}}_{22}^{32}$$

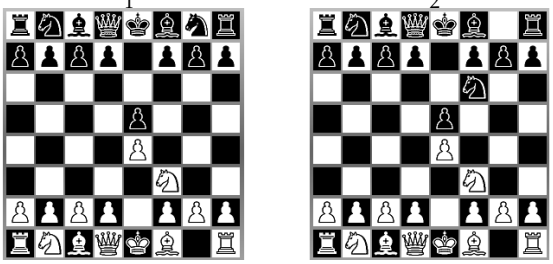
$$b_3^2 = \left\{ \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{22}^{32} \right\}_{46}^{16} \quad = \quad \boxed{\boxed{\boxed{b_0}_{37}^{6}}_{22}^{32}}_{46}^{16}$$

$$b_4^2 = \left\{ \left\{ \left\{ \left\{ b_0 \right\}_{37}^{6} \right\}_{22}^{32} \right\}_{46}^{16} \right\}_{29}^{22} \quad = \quad \boxed{\boxed{\boxed{\boxed{b_0}_{37}^{6}}_{22}^{32}}_{46}^{16}}_{29}^{22}$$

Figure 9. Board sequence z_2' corresponding to the moves (1) e4 Kf6; (2) Kf3 e5.



b_1^3 b_2^3



b_3^3 b_4^3

$$b_1^3 = \left\{ b_0 \right\}_{46}^{16} \quad = \quad \boxed{b_0}_{46}^{16}$$

$$b_2^3 = \left\{ \left\{ b_0 \right\}_{46}^{16} \right\}_{29}^{13} \quad = \quad \boxed{\boxed{b_0}_{46}^{16}}_{29}^{13}$$

$$b_3^3 = \left\{ \left\{ \left\{ b_0 \right\}_{46}^{16} \right\}_{29}^{13} \right\}_{37}^{6} \quad = \quad \boxed{\boxed{\boxed{b_0}_{46}^{16}}_{29}^{13}}_{37}^{6}$$

$$b_4^3 = \left\{ \left\{ \left\{ \left\{ b_0 \right\}_{46}^{16} \right\}_{29}^{13} \right\}_{37}^{6} \right\}_{22}^{32} \quad = \quad \boxed{\boxed{\boxed{\boxed{b_0}_{46}^{16}}_{29}^{13}}_{37}^{6}}_{22}^{32}$$

Figure 10. Board sequence z_3' corresponding to the moves (1) Kf3 e5; (2) e4 Kf6.

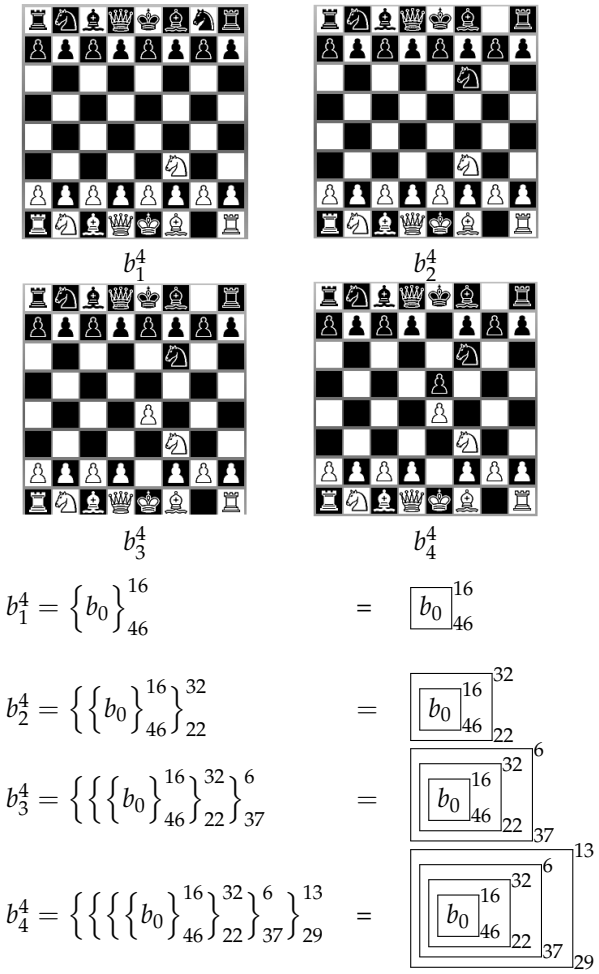


Figure 11. Board sequence z_4' corresponding to the moves (1) Kf3 e5; (2) e4 Kf6.

The primitive recursive function $F_{18}(\cdot)$ in (89) takes a game G_z, a G-number EM of moves such that $(\forall\, m \in EM)\,\{\mathtt{pcb}(m) = \mathtt{tlb}(z)\,\}$, a G-number Z of games and a number t. This function computes the continuation $[z \otimes_r [\mathtt{scb}((EM)_{s(t)})]$ of G_z and appends it at the right of the games Z computed thus far.

The primitive recursive function $F_{17}(\cdot)$ in (88) takes a game G_z, a G-number M of moves such that $(\forall\, m \in EM)\,\{\mathtt{pcb}(m) = \mathtt{tlb}(z)\,\}$ and returns the G-number of games each of which is a continuation of G_z with each successor board of the moves EM.

The primitive recursive function $F_{16}(\cdot)$ in (87) takes a game G_z and the G-number EM of moves of epoch $t > 0$ and computes the G-number of all games that extend G_z with the appropriate successor boards in EM (i.e., with the successor board of each move $m \in EM$ such that $\mathtt{pcb}(m) = \mathtt{tlb}(z)$).

The primitive recursive function $F_{15}(\cdot)$ in (86) takes a G-number of games Z and $t \geq 1$ and computes the G-number of games by extending every game in Z with all appropriate successor boards in EM_{t+1}. If $t > 0$, then

$$\mathbb{G}_t = \otimes_r|_{i=1}^{t}[J(i)] \tag{90}$$

is the G-number of all games up to epoch t. All games in $\mathbb{G}_t$ are *historical* in the sense that they are based on the moves in each epoch E_t of $\mathbb{H}_t$ in (80). All games in $\mathbb{G}_t$ are *legal*, because they are obtained by legal moves from the initial board b_0. We have the following lemma.

Lemma 4. *Let $G_z \in \mathbb{G}_t$, $t > 0$, and $b = \mathtt{tlb}(G_z)$. Then $\mathtt{Lt}(z) = t + 1$ and if t is even, then player 1 is to play on b; if t is odd, player 2 is to play on b.*

Let

$$P_\Gamma(z) \equiv \mathtt{hgm}(z) \wedge G_z \in \mathbb{G}_{\mathtt{Lt}(z)-1} \tag{91}$$

be a primitive recursive predicate that holds of z if G_z is a chess game and let

$$\Gamma = \{z \in \mathbb{N} | P_\Gamma(z)\} \tag{92}$$

be the set of numbers for which $P_\Gamma(\cdot)$ holds. Since every primitive recursive function is computable (See Chapter 3 in [5]), we have the following lemma.

Lemma 5. *The set of chess games* Γ *is recursive.*

Lemma 5 shows that the characteristic of chess game historicity or, equivalently, chess game legality is pr-decidable.

Let G_z be a game and let z' be a sequence of boards such that $\mathtt{Lt}(z') > 1$. We say that z' is a *subgame* of G_z if the primitive recursive predicate $\mathtt{sbg}(\cdot)$ in (93) holds of G_z and z'.

$$\begin{array}{l} \mathtt{sbg}(z', G_z) \equiv \\ \{1 < \mathtt{Lt}(z') \leq \mathtt{Lt}(z) \wedge \\ (\exists i \in \mathtt{ggn}(1, \mathtt{Lt}(z) \dot{-} 1, 1))\{ \\ (\forall k \in \mathtt{ggn}(1, \mathtt{Lt}(z'), 1))\{(z')_k = (z)_{i+k-1}\}\}\} \end{array} \tag{93}$$

In other words, z' must have at least 2 boards but no more boards than z and be a subsequence of z that starts at some position i between 1 and $\mathtt{Lt}(z') \dot{-} 1$. Consider Figure 10. If

$$z_3 = [b_0, b_1^3, b_2^3, b_3^3, b_4^3],$$

then

$$\begin{array}{lcl} \mathtt{sbg}([b_0, b_1^3], G_{z_3}) & = & 1; \\ \mathtt{sbg}([b_1^3, b_2^3, b_3^3], G_{z_3}) & = & 1; \\ \mathtt{sbg}([b_1^3, b_3^3, b_4^3], G_{z_3}) & = & 0; \\ \mathtt{sbg}([b_2^3, b_4^3], G_{z_3}) & = & 0. \end{array}$$

We define the primitive recursive predicate $\mathtt{sbg}_2(\cdot)$ to hold when z' starts at a specific board i in G_z.

$$\begin{array}{l} \mathtt{sbg}_2(z', G_z, i) \equiv \\ \{1 < \mathtt{Lt}(z') \leq \mathtt{Lt}(z) \wedge \{0 < i \wedge \\ (\forall k \in \mathtt{ggn}(1, \mathtt{Lt}(z'), 1))\{(z')_k = (z)_{i+k-1}\}\}\} \end{array} \tag{94}$$

Thus,

$$\mathtt{sbg}_2([b_0, b_1^3], G_{z_3}, 1) = \mathtt{sbg}_2([b_2^3, b_3^3, b_4^3], G_{z_3}, 3) = 1.$$

4.13. Repetition and 50-Move Rule

We now return to the remaining two types of draw: draw by repetition and the 50-move rule, which we promised to consider at the end of Section 4.9. We recall that a draw by repetition is achieved when the same board position occurs three times in a row with the same player to move. Consider the sequence of boards in Figure 12. Board b_0 occurs three times in this sequence (i.e., $b_0 = b_4 = b_8$) with player 1 to move.

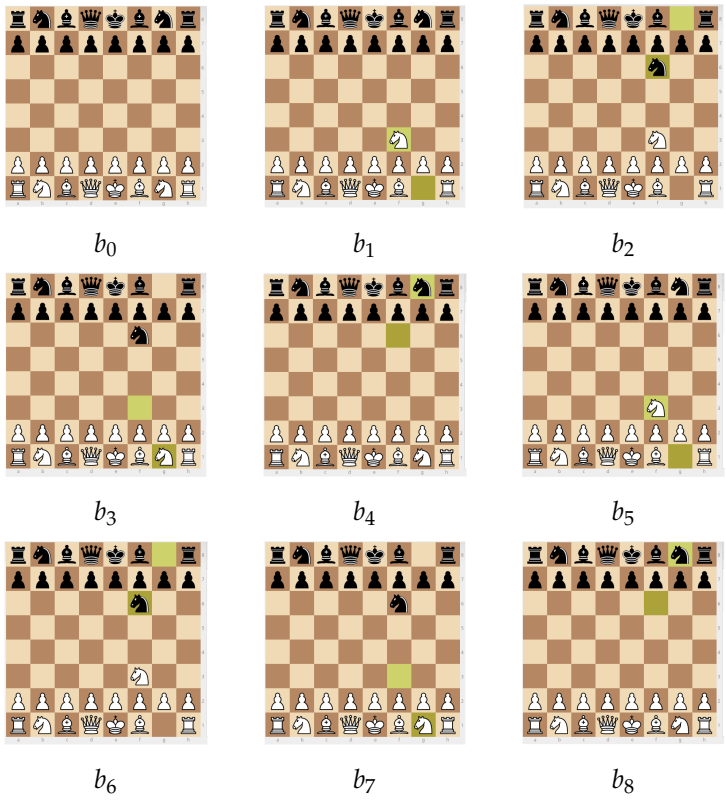


Figure 12. Draw by threefold repetition.

Let z' be a subgame with player x to play on $(z')_1$. The same board occurs in a row at $(z')_5$ since x and $\bar{x}$ must make one move each and then reverse that move. Player x then plays on $(z')_5$ and the same board occurs in a row at $(z')_9$, since x and $\bar{x}$ make one move each again and reverse it. Thus, z' is a draw by repetition if it has 9 boards such that $(z')_1 = (z')_5 = (z')_9$. In other words, if

$$\mathtt{Lt}(z') = 9 \wedge (z')_1 = (z')_5 = (z')_9.$$

Let $G_z \in \mathbb{G}_t$, $t > 0$, then G_z is a draw by repetition if

$$\mathtt{drp}(G_z) \equiv \{\mathtt{Lt}(z) \geq 9 \wedge \mathtt{tlb}(z) = (z)_{\mathtt{Lt}(z) \dot{-} 8} = (z)_{\mathtt{Lt}(z) \dot{-} 4}\} \tag{95}$$

The 50-move rule states that a game is a draw when the last 50 moves contain no capture or pawn move. If a game G_z is a draw by the 50-move rule, then $G_z \in \mathbb{G}_t$, $t \geq 49$. Let z' be a subgame of G_z that has 50 boards (i.e., $\mathtt{Lt}(z') = 50$). Then G_z is a 50-move rule draw if for all boards in z' the pawn positions are same and the counts of all white and black pieces remain constant.

We define several auxiliary primitive recursive predicates to compose them into a primitive recursive predicate that holds of G_z if the latter is a draw by the 50-move rule. Let

$$\begin{aligned} \zeta_{\mathtt{wpn}} &= \mathtt{ggn}(2, 9, 1); \\ \zeta_{\mathtt{bpn}} &= \mathtt{ggn}(18, 25, 1) \end{aligned}$$

be the G-numbers of the white and black pawns, respectively. The primitive recursive predicates $\mathtt{wpx50}(\cdot)$ and $\mathtt{bpx50}(\cdot)$ below hold if every white and black pawn is present or abscent on both boards (i.e., b_1 and b_2) and, if any white or black pawn is present on both boards, its position is the same on both.

$$\begin{aligned}&\texttt{wpx50}(b_1,b_2)\equiv\\&\{(\forall i\in\zeta_{\texttt{wpn}})\{\{i\in b_1\wedge i\in b_2\}\vee\ \{i\notin b_1\wedge i\notin b_2\}\}\}\wedge\\&\{(\forall i\in\zeta_{\texttt{wpn}})\{\{i\notin b_1\vee i\notin b_2\}\vee\ \texttt{cpsn}(i,b_1)=\texttt{cpsn}(i,b_2)\}\}\end{aligned}\tag{96}$$

$$\begin{aligned}&\texttt{bpx50}(b_1,b_2)\equiv\\&\{(\forall i\in\zeta_{\texttt{bpn}})\{\{i\in b_1\wedge i\in b_2\}\vee\ \{i\notin b_1\wedge i\notin b_2\}\}\}\wedge\\&\{(\forall i\in\zeta_{\texttt{bpn}})\{\{i\notin b_1\vee i\notin b_2\}\vee\ \texttt{cpsn}(i,b_1)=\texttt{cpsn}(i,b_2)\}\}\end{aligned}\tag{97}$$

We recall that ζ_{cp} in (41) defines the G-number of all the white and black pieces. The primitive recursive predicate

$$\texttt{cntx50}(b_1,b_2)\equiv(\forall j\in\zeta_{\texttt{cp}})\{\texttt{cnt}(j,b_1)=\texttt{cnt}(j,b_2)\}$$

holds if the counts of all pieces on the boards b_1 and b_2 are equal.

Let z be a sequence of boards. The primitive recursive predicate

$$\texttt{wbp50}(z)\equiv(\forall t\in\texttt{ggn}(1,\texttt{Lt}(z)\dot{-}1,1))\{\texttt{wpx50}((z)_t,(z)_{t+1})\wedge\texttt{bpx50}((z)_t,(z)_{t+1})\}$$

holds if the predicates $\texttt{wpx50}(\cdot)$ and $\texttt{bpx50}(\cdot)$ hold for every pair of consecutive boards in z. The primitive recursive predicate

$$\texttt{cnt50}(z)\equiv(\forall t\in\texttt{ggn}(1,\texttt{Lt}(z)\dot{-}1,1))\{\texttt{cntx50}((z)_t,(z)_{t+1})\}$$

holds if the counts of all pieces on all boards in z are equal. The primitive recursive predicate

$$\texttt{drx50}(z)\equiv\texttt{wbp50}(z)\wedge\texttt{cnt50}(z)$$

holds if the boards in z have no pawn moves or piece captures.

The game G_z is a draw by the 50-move rule if there is a position i in z that starts a subgame of length 50 for which $\texttt{drx50}(\cdot)$ holds. The primitive recursive predicate

$$\texttt{dr50}(G_z)\equiv\texttt{Lt}(z)\geq 50\wedge(\exists t\in\texttt{ggn}(1,\texttt{Lt}(z)\dot{-}49,1))\{\texttt{drx50}(\texttt{ssq}(z,t,t+49)\}\tag{98}$$

holds if G_z is a draw by the 50-move rule.

4.14. Classification of Games

We define the primitive recursive predicate

$$\texttt{drg}(G_z)\equiv\texttt{drp}(G_z)\vee\texttt{dr50}(G_z)\vee\texttt{bsm}\Big(\texttt{tlb}(G_z)\Big)\vee\texttt{wsm}\Big(\texttt{tlb}(G_z)\Big)\vee\texttt{dp}\Big(\texttt{tlb}(G_z)\Big)\tag{99}$$

to hold when G_z is a draw game by repetition, the 50-move rule, stalemate, or dead position. The primitive recursive predicate

$$\texttt{wng}(G_z,x)\equiv\begin{cases}\texttt{bkm}\Big(\texttt{tlb}(G_z)\Big) & \text{if } x=1,\\ \texttt{wkm}\Big(\texttt{tlb}(G_z)\Big) & \text{if } x=2,\\ 0 & \text{otherwise}\end{cases}\tag{100}$$

holds when G_z ends in a mate for $\bar{x}$. The primitive recursive predicate

$$\texttt{ufg}(G_z,x)\equiv\neg\texttt{drg}(G_z)\wedge\neg\texttt{wng}(G_z,x)\tag{101}$$

holds when G_z is unfinished for x (i.e., it is neither a draw nor a win for x).

By Lemma 4, player 1 can win only those games in $\mathbb{G}_t$ if t is odd and player 2 can win only games in $\mathbb{G}_t$ if t is even. Let $G_z\in\mathbb{G}_t$, $t>0$, and let x be the player to play at t. If G_z is not a win for $\bar{x}$, then x can win in 1, 3, 5, ..., $2k+1$ moves. Let n be an odd number of moves into the future. We inquire if it is possible to find games $G_y\in\mathbb{G}_{t+n}$ that

are continuations of G_z and are wins for x in a primitive recursive fashion. The primitive recursive predicate

$$\mathtt{cng}(y,z) \equiv z = \mathtt{ssq}(y,1,\mathtt{Lt}(z)) \wedge P_\Gamma(y) \tag{102}$$

determines if G_y is a continuation of G_z. For any G_z, we let the primitive recursive predicate

$$\mathtt{wcng}(y,x) \equiv \mathtt{cng}(y,z) \wedge \mathtt{wng}(G_y,x)$$

hold when G_y is a continuation of G_z that is a win for x. Let $\mathtt{wnx}(y) \equiv \mathtt{wcng}(y,1)$ and $\mathtt{bnx}(y) \equiv \mathtt{wcng}(y,2)$. If $G_z \in \mathbb{G}_t$, $t > 0$, then the primitive recursive function

$$\mathtt{wngs}(G_z,x,t,n) = \begin{cases} \mathtt{map}_{\mathtt{wnx}}(\mathbb{G}_{t+n}) & \text{if } 2 \nmid n \wedge x = 1, \\ \mathtt{map}_{\mathtt{bnx}}(\mathbb{G}_{t+n}) & \text{if } 2 \nmid n \wedge x = 2, \\ 0 & \text{otherwise} \end{cases} \tag{103}$$

returns the G-number of all continuations of G_z in $\mathbb{G}_{t+n}$ that are wins for x whose turn it is to play on the tail of G_z. The primitive recursive predicate

$$\mathtt{dcng}(y) \equiv \mathtt{cng}(y,z) \wedge \mathtt{drg}(G_y)$$

holds when G_y is a continuation of G_z that is a draw. If $G_z \in \mathbb{G}_t$, $t > 0$, then the primitive recursive function

$$\mathtt{drgs}(G_z,t,n) = \begin{cases} \mathtt{map}_{\mathtt{dcng}}(\mathbb{G}_{t+n}) & \text{if } 2 \nmid n, \\ 0 & \text{otherwise} \end{cases} \tag{104}$$

returns the G-number of all continuations of G_z in $\mathbb{G}_{t+n}$ that are draws for x whose turn it is to play on the last board of G_z. The primitive recursive predicate

$$\mathtt{ucng}(y,x) \equiv \mathtt{cng}(y,z) \wedge \mathtt{ufg}(G_y,x)$$

holds when G_y is a continuation of G_z that is unfinished for x. Let $\mathtt{wux}(y) \equiv \mathtt{ucng}(y,1)$ and $\mathtt{bux}(y) \equiv \mathtt{ucng}(y,2)$. If $G_z \in \mathbb{G}_t$, $t > 0$, then the primitive recursive function

$$\mathtt{ufgs}(G_z,x,t,n) = \begin{cases} \mathtt{map}_{\mathtt{wux}}(\mathbb{G}_{t+n}) & \text{if } 2 \nmid n \wedge x = 1, \\ \mathtt{map}_{\mathtt{bux}}(\mathbb{G}_{t+n}) & \text{if } 2 \nmid n \wedge x = 2, \\ 0 & \text{otherwise} \end{cases} \tag{105}$$

returns the G-number of all continuations of G_z in $\mathbb{G}_{t+n}$ that are unfinished for x whose turn it is to play on the tail of G_z.

Let $G_z \in \mathbb{G}_t$, $t > 0$, and let x be the player whose turn it is to play on the tail of G_z. We will further assume that G_z is unfinished (i.e., $\mathtt{ufg}(G_z,x) = 1$) for x. We call G_z *winnable* for x in n moves if $\mathtt{wngs}(G_z,x,t,n) \neq 0$, *drawable* for x in n moves if $\mathtt{drgs}(G_z,t,n) \neq 0$, and *unfinishable* for x in n moves if $\mathtt{ufgs}(G_z,x,t,n) \neq 0$. We will call unfinishable games *hangs*. If a player makes a move that results in a hang, we will say that the player *hangs* the game or that the game is *hung*. We have the following lemma.

Lemma 6. *Let $G_z \in \mathbb{G}_t$, $t > 0$, be a hang and x be the player whose turn it is to play on $b = \mathtt{tlb}(z)$. If n is an odd positive integer, then it is pr-decidable whether G_z is is winnable for x within n moves, is drawable for x within n moves, or is a hang for x within n moves.*

Of course, if a game is winnable, drawable, or unfinishable for x, x may still lose it within n moves, because there may be a sequence of at most n moves from the tail of G_z that allows $\bar{x}$ to win.

We pose the question if it is pr-decidable to whether G_z is *absolutely winnable* for x within n moves when x is to play on the tail of G_z. G_z is absolutely winnable for x if any

continuation of G_z of less than or equal to n boards results in a win for x and is not winnable or drawable for $\bar{x}$.

Let us assume that $G_z \in \mathbb{G}_t$, $t > 0$, and x is the player whose turn it is to play on the tail of G_z and G_z is unfinished (i.e., $\mathtt{ufg}(G_z, x) = 0$). Let

$$\begin{array}{l} g_1 = \mathtt{ggn}(t+1, t+n, 2) \\ g_2 = \mathtt{ggn}(t+2, t+n, 2) \end{array} \tag{106}$$

be the G-numbers of game epochs strictly between t and $t+n$, where $g_1 = [(t+1), (t+1)+2, \ldots, (t+1)+2j]$, and $g_2 = [t+2, t+2 \cdot 2, \ldots, t+2k]$, where $(t+1)+2j$ and $t+2k$ are the largest numbers less than $t+n$ obtained in even increments from $(t+1)$ and $(t+2)$, respectively. G-number g_1 contains the numbers of epochs, where $\bar{x}$ plays and x can win, draw, or hang, and g_2 contains the numbers of epochs where x plays and $\bar{x}$ can win, draw, or hang.

The primitive recursive predicate

$$\mathtt{awg}_1(G_z, x, n) \equiv (\forall t \in g_1)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \mathtt{wcng}(y, x)\}$$

holds if x wins every continuation G_y of G_z within n moves and the primitive recursive predicate

$$\mathtt{awg}_2(G_z, x, n) \equiv (\forall t \in g_2)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \mathtt{ucng}(y, \bar{x})\}$$

holds if every continuation G_y of G_z within n moves is a hang for $\bar{x}$.

The primitive recursive predicate

$$\mathtt{awg}(G_z, x, n) \equiv \mathtt{awg}_1(G_z, x, n) \wedge \mathtt{awg}_2(G_z, x, n) \tag{107}$$

holds if G_z is absolutely winnable for x within n moves.

Is it pr-decidable if x can win or draw G_z from its tail within the next n moves while $\bar{x}$ cannot win under any sequence of moves less than or equal to n? If G_z is such a game, we call it *favorable* for x.

We define the primitive recursive predicate

$$\mathtt{fvg}_1(G_z, x, n) \equiv (\forall t \in g_1)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \{\mathtt{dcng}(y) \vee \mathtt{wcng}(y, x)\}\}$$

to hold if x wins or draws in every continuation G_y of G_z within n moves and define the primitive recursive predicate

$$\mathtt{fvg}_2(G_z, x, n) \equiv (\forall t \in g_2)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \mathtt{ucng}(y, \bar{x})\}$$

to hold if every continuation G_y of G_z within n moves is a hang for $\bar{x}$.

The primitive recursive predicate

$$\mathtt{fvg}(G_z, x, n) \equiv \mathtt{fvg}_1(G_z, x, n) \wedge \mathtt{fvg}_2(G_z, x, n) \tag{108}$$

holds if G_z is favorable for x within n moves in that in any continuation G_y of G_z within the next n moves x wins or draws whereas $\bar{x}$ hangs.

Is it pr-decidable whether neither x nor $\bar{x}$ can win or draw G_z within the next n moves from the tail of G_z? In other words, is any continuation of G_z within the next n moves from its tail is a hang? If that it the case, let us refer to G_z as an *absolute hang* or *absolutely unfinishable*.

The primitive recursive predicate

$$\mathtt{ahg}_1(G_z, x, n) \equiv (\forall t \in g_1)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \mathtt{ucng}(y, x)\}$$

holds if x hangs in every continuation G_y of G_z within n moves, and the primitive recursive predicate

$$\mathtt{ahg}_2(G_z, x, n) \equiv (\forall t \in g_2)(\forall y \in \mathbb{G}_t)\{\neg \mathtt{cng}(y, z) \vee \mathtt{ucng}(y, \bar{x})\}$$

holds if every continuation G_y of G_z within n moves is a hang for $\bar{x}$.

The primitive recursive predicate

$$\mathtt{ahg}(G_z, x, n) \equiv \mathtt{ahg}_1(G_z, x, n) \wedge \mathtt{ahg}_2(G_z, x, n) \tag{109}$$

holds if G_z is unfinishable for x and $\bar{x}$ within n moves.

We summarize the above arguments in the following lemma.

Lemma 7. *Let $G_z \in \mathbb{G}_t$, $t > 0$, be a hang and x be the player whose turn it is to play on $b = \mathtt{tlb}(z)$. If n be an odd positive integer, then it is pr-decidable whether G_z is absolutely winnable for x within n moves, is favorable for x within n moves, or is an absolute hang for x (and $\bar{x}$) within n moves.*

Before we leave this section, we summarize our findings in the following lemma.

Lemma 8. *The following characteristics of chess are pr-decidable: board validity, checkmate, stalemate, draw, potential and actual reachability, game winnability, drawability, and unfinishability within the next n moves, absolute game winnability and unfinishability within the next n moves and game favorability within the next n moves, where n is a positive odd integer.*

5. Procedures

Suppose $G_z \in \mathbb{G}_t$, $t > 0$, and the player x is to play on the tail of G_z. Is an optimal continuation of G_z for x within n moves pr-computable for some positive odd n? An optimal continuation may be the continuation of 0 moves (i.e., resignation), which is the only rational decision in the absence of any wins or draws. Lemma 8 suggests a host of primitive recursive procedures for x to find an optimal continuation including the resignation. If $\mathtt{awg}(G_z, x, n)$ is true, then let

$$Z_a = \mathtt{wngs}(G_z, x, t, n)$$

be the pr-computable G-number of winnable games for x within n moves, which, since G_z is absolutely winnable, includes all continuations of G_z within n moves. The next move from $(Z_a)_1$ can be arbitrarily chosen for x to play (e.g., $((Z_a)_1)_{\mathtt{Lt}(z)+1}$). Alternatively, the minimalization can be used to find the index of $z' \in Z_a$ such that

$$(\forall z'' \in Z_a)\{z' = z'' \vee \mathtt{Lt}(z') \leq \mathtt{Lt}(z'')\}.$$

Since the above predicate is primitive recursive, so is its minimalization. If $G_{z'}$ is the shortest absolutely winnable game found through the minimalization of the above primitive recursive predicate, the next move is chosen as

$$(z')_{\mathtt{Lt}(z)+1}.$$

If $\mathtt{awg}(G_z, x, n) = 0$ but $\mathtt{fvg}(G_z, x, n) = 1$, then let

$$\begin{aligned} Z_w &= \mathtt{wngs}(G_z, x, t, n); \\ Z_d &= \mathtt{drgs}(G_z, x, t, n). \end{aligned}$$

Since G_z is favorable, all continuations of G_z within the next n moves will be in Z_w or Z_d. If $Z_w \neq 0$, x uses the minimalization to find the shortest continuation in Z_w as outlined above and selects the next move from that continuation. If $Z_w = 0$, then all continuations of G_z are draws for x within the next n moves, and x can choose the next move from the

shortest continuation in Z_d. If $\mathtt{awg}(G_z, x, n) = \mathtt{fvg}(G_z, x, n) = 0$ but $\mathtt{ahg}(G_z, x, n) = 1$, then x chooses the next move from the shortest z in

$$Z_h = \mathtt{ufgs}(G_z, x, t, n).$$

What if $Z_w = Z_d = Z_h = 0$? In this case, G_z is *absolutely losable* or an *absolute loss* for x within n moves. The primitive recursive predicate

$$\begin{array}{rcl} \mathtt{alg}(G_z, x, n) & \equiv & \mathtt{wngs}(G_z, x, \mathtt{Lt}(z) + 1, n) \\ & = & \mathtt{drgs}(G_z, x, \mathtt{Lt}(z) + 1, n) \\ & = & \mathtt{ufgs}(G_z, x, \mathtt{Lt}(z) + 1, n) \\ & = & 0 \end{array} \tag{110}$$

holds when G_z is absolutely losable for x within the next n moves. In this case, the concept of lookahead (i.e., $n + m$ moves ahead for some positive even number m) can be applied and the above primitive recursive steps of computing Z_w, Z_d, and Z_h repeated so long as $n + m \leq K$, where K is some arbibrary large positive odd integer. Taking $M = 399$ should suffice in light of the fact that the longest game so far in the history of chess took 269 moves to complete and lasted 20 h and 15 min [8]. If G_z remains absolutely losable for x within the next K moves, x resigns.

What if $Z_h \neq 0$ (i.e., there are absolute hangs)? We can use the same incremental lookahead method to determine if any $G_{z'} \in Z_h$ is absolutely winnable or favorable for x within the next K moves. If there is an absolutely winnable game, x plays the first move of that game. If there is a favorable game, x plays the first move of that game.

If there are no absolutely winnable or favorable games within the next K moves for any game in Z_h, it must be decided for x which unfinishable game in Z_h to choose. One possible pr-computable procedure is to choose a continuation $G_{z'}$ of G_z in Z_h for which the counts of the pieces for each board of $G_{z'}$ are equal. If there is such a game, x plays its first move.

Another, more flexible, pr-computable procedure is to assign an arbitrary value (a positive integer) to each piece: 2 to a pawn, 3 to a knight and a bishop, 5 to a rook, 7 to a queen and compute the value of each board for player x as the sum of x's pieces on the board. Then a continuation $G_{z'} \in Z_h$ of G_z is chosen where each board from the tail of G_z to the tail of $G_{z'}$ has the same value for x and $\bar{x}$ or the difference in the values does not exceed an arbitrarily chosen number. In general, any decision procedure or a utility function that is built from primitive recursive functions and predicates that compute properties of G-numbers will be primitive recursive

The above discussion furnishes us with the following theorem and a corollary.

Theorem 1. *Let $G_z \in \mathbb{G}_t$, $t > 0$, be a hang and x be the player whose turn it is to play on $b = \mathtt{tlb}(z)$ and let K be a large positive odd integer. Then there exist pr-computable procedures to compute optimal continuations of G_z for x and to decide if G_z is absolutely losable for x within the next M moves.*

Corollary 1. *Absolute losability of a chess game is a pr-decidable characteristic.*

6. Discussion

In the AI game playing literature (e.g., [9,10]), game engines are functions computed by programs that can play specific games by generating legal moves against humans or other programs with varying degrees of success. These programs almost never compute total functions, because they are based on search control heuristics (e.g., A*, minimax, alpha-beta pruning, etc.). Thus, even when their inputs and outputs can be represented as natural numbers with a rigorous Gödel numbering scheme, they can be construed only as partially computable functions. Furthermore, programs that play games with difficult combinatorics (chess, checkers, Go, etc.) break the fundamental tenet of classical computability in Rogers' quote from [4] in Section 1, because, to be efficient, they must esti-

mate, ahead of computation, how long the decision making process takes and heuristically prune continuations.

In the classification of Barr and Feigenbaum [9], D2PBGs are games that can be represented with AND/OR trees. Specifically, Barr and Feigenbaum write that

> ‘‘[a]t each turn, the rules define both what moves are legal and the effect that each possible move will have; there is no element of chance. In contrast to card games in which the players’ hands are hidden, each player has complete information about his opponent’s position, including the choices open to him and the moves he has made. The game begins from a specified state, often a configuration of men on a board. It ends in a win for one player and a loss for the other, or possibly in a draw.’’

Russell and Norvig [10] characterize chess as “a game of perfect information”, because “the agent can perceive everything there is to know about the environment”. In our opinion, Russell and Norvig’s definition is less precise than Barr and Feigenbaum’s, because it appears to separate computation and perception. If perception is not computation, then AI D2PBG engines fall outside of the scope of classical computability. If perception is computation, many such engines are partially computable functions, because they are not total due to heuristic pruning.

AND/OR game trees can be searched with minimax [9]. If in a chess engine both players use minimax on the complete AND/OR game tree, which is feasible only in theory, and if the board evaluation function applied to each tree node by each player is primitive recursive, then the engine is a primitive recursive function by Lemma 8 and the Section 5 theorem. However, if the board evaluation function in minimax is *not* primitive recursive (e.g., if it is computable but not primitive recursive or partially computable), then the engine is not primitive recursive.

What if we put aside heuristic search control (e.g., minimax) and consider a chess game engine that uses an artificial or convolutional neural network (ANN or ConvNet) (e.g., [11])? In this case, the chess engine computes a primitive recursive function if and only if the synapse weights are natural numbers, which is never the case in the state-of-the-art ANNs or ConvNets that play such games as chess and Go. They are deep multi-layer networks where each weight is a real number (typically, but not always, between 0 and 1). But, since there are uncountably many reals between 0 and 1 or in any interval defined by two distinct real numbers, it is impossible to generate network states with primitive recursion. Consequently, if the chess engine is a ANN/ConvNet, then the engine may be computable, but not primitive recursive. Of course, even the computability of the engine depends on whether, in addition to representing inputs and outputs as natural numbers, we are able to show that the engine computes a total function.

Any characteristic of a D2PBG game with a finite history (e.g., Tic Tac Toe) is pr-decidable and pr-computable in the sense that the entire game history is a G-number and an optimal sequence of moves for either player can be found using number-theoretic methods from an arbitrary board within that number.

A D2PBG all of whose boards can be calculated in a primitive recursive fashion within the next K moves for some arbitrarily large number K (with the magnitude of K dependent on the rules of the game) is pr-decidable for K in the sense that for either player it is possible to decide in a primitive recursive fashion whether an unfinished game is absolutely winnable, favorable, or absolutely losable within K moves. If a game is absolutely unfinishable within the next K moves, then there exist primitive recursive calculation procedures to choose the next move and in that sense (and that sense only) chess is pr-computable.

As we showed in this article, some characteristics of chess are pr-computable. Any characteristic of the chess board (and, in general, a board in a D2PBG) represented as a G-number that can be decided by a primitive recursive predicate through primitive recursive

number-theoretic methods is pr-decidable. Since sequences of boards can be combined into G-numbers as games, any characteristic of the G-number of a game that can be calculated through primitive recursive number-theoretic methods is pr-computable and pr-decidable.

The following questions remain open with respect to chess as a D2PBG:

1. Is chess a D2PBG with a finite history? In other words, is there $t > 0$ such that $(\forall t')\{t' \leq t \vee E_{t'} = 0\}$? (See Equation (79)).
2. Are there computable decision procedures for player x that are not primitive recursive and that guarantee for x to win or draw an unfinished game if $\bar{x}$ uses only primitive recursive decision procedures?
3. Are there games that x cannot lose when x uses only primitive recursive decision procedures while $\bar{x}$ uses computable decision procedures that are not primitive recursive?

7. Summary

We investigated several characteristics of chess with methods of computability and number theories. We showed that chess boards can be represented as Gödel numbers (G-numbers). The following characteristics were shown to be pr-decidable: board validity, position reachability, potential and actual position reachability, check, mate, stalemate, and draw.

We showed that it is pr-decidable whether an unfinished game is absolutely winnable, favorable (i.e., winnable or drawable), or absolutely losable within a specified number of moves for the player whose turn it is to play on the last board. We also showed that that there exist primitive recursive procedures to compute optimal continuations of an unfinished game and that the set of all chess games is recursive.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math10071016/s1.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Bolon, T. *How to Never Lose at Tic Tac Toe*; Book Country: New York, NY, USA, 2013.
2. Daly, W., Jr. Computer Strategies for the Game of Qubic. Master's Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, 1961.
3. Kleene, S.C. *Introduction to Metamathematics*; D. Van Nostrand: New York, NY, USA, 1952.
4. Rogers, H., Jr. *Theory of Recursive Functions and Effective Computability*; The MIT Press: Cambridge, MA, USA, 1988.
5. Davis, M.; Sigal, R.; Weyuker, E. *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, 2nd ed.; Harcourt, Brace & Company: Boston, MA, USA, 1994.
6. Kulyukin, V. On Primitive Recursiveness of Tic Tac Toe. In Proceedings of the International Conference on Foundations of Computer Science (FCS'19), Las Vegas, NV, USA, July 29–1 August 2019; pp. 9–15,
7. Meyer, M.; Ritchie, D. The Complexity of Loop Programs. In Proceedings of the ACM National Meeting, Washington, DC, USA, 1967; pp. 465–469. Available online: https://people.csail.mit.edu/meyer/meyer-ritchie.pdf (accessed on 11 February 2022).
8. The Longest Tournament Chess Game. Available online: https://www.chess.com/blog/ThummimS/the-longest-tournament-chess-game (accessed on 21 January 2022).
9. Barr, A.; Feigenbaum, E. *The Handbook of Artificial Intelligence*; Addison-Wesley: Reading, MA, USA, 1982; Volume 1.
10. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson: Hoboken, NJ, USA, 1995.
11. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.