UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DOCTORADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

DETERMINACIÓN DE LOS ATRIBUTOS PARTICULARES DE LOS PATRONES QUE DAN VENTAJA EN LOS JUEGOS DE SUMA-CERO. ATRIBUTOS Y SIMILITUDES

TESIS
QUE PARA OPTAR POR EL GRADO DE DOCTOR
EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
MANUEL CRISTÓBAL LÓPEZ MICHELONE

TUTOR PRINCIPAL: DR. JORGE LUIS ORTEGA ARJONA
TUTORES. DR. HÉCTOR BENÍTEZ, DR. VLADISLAV KHARTCHENKO
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

CIUDAD UNIVERSITARIA, CD. MX., SEPTIEMBRE 2021

Determinación de los atributos particulares de los patrones que dan ventaja en los juegos de suma–cero. Atributos y similitudes

Manuel Cristóbal López Michelone

Índice general

Índice de figuras

"El ajedrez no es un juego. El ajedrez es una forma bien definida de computación. Puede que no te sea posible concebir las respuestas; pero en teoría debe existir una solución, un procedimiento exacto en cada posición. Ahora bien, los juegos verdaderos no son así. La vida real no es así. La vida real consiste en farolear, en tácticas pequeñas y astutas, en preguntarse uno mismo qué será lo que el otro hombre piensa que yo entiendo hacer. Y en esto consisten los juegos en mi teoría".

John von Neumann[1]

[1]Bronowski, J. (1973/1979). El ascenso del hombre. Trad. Alejandro Ludlow Wiechers, Francisco Rebolledo López, Víctor M. Lozano, Efraín Hurtado y Gonzalo González Fernández. Londres/Bogotá: BBC/Fondo Educativo Interamericano.

Capítulo 1

Introducción

1.1. El contexto

La teoría de juegos es una rama de las matemáticas que estudia las interacciones entre individuos que toman decisiones, es decir, es una teoría del conflicto –todo esto en un marco de incentivos– lo cual en términos generales se denomina "juego" [62].

Una sub-rama de esta ciencia se define como juegos de suma–cero (o juegos de suma nula) y de información perfecta. El primer caso se refiere a modelos en los que la ganancia de un jugador implica necesariamente una pérdida de otro exactamente del mismo valor [69], y en el segundo caso, al desarrollarse estos juegos, en cualquiera de las partidas, cada jugador tiene conocimiento de todas las elecciones que le precedieron, así como de los jugadores que las realizaron en el momento de hacer cualquier de sus elecciones [73].

Se dice que un juego tiene solución final o está resuelto cuando se puede encontrar la estrategia óptima para el mismo, en donde uno de los dos jugadores siempre puede forzar la victoria (o el empate). Todo esto de acuerdo con el *Teorema de Zermelo* [53, 75][1].

La ciencia de la computación se ha interesado por investigar los juegos de

[1] Así llamado en honor de Ernst Zermelo, que asegura que en cualesquiera juegos finitos entre dos personas en los cuales los jugadores hacen movimientos alternadamente y en donde el azar no afecta la toma de decisiones, si el juego no puede acabar en un empate, entonces debe existir hipotéticamente una estrategia óptima ganadora para alguno de ellos.

suma–cero[2], los conflictos que se generan en ellos y las soluciones posibles de los mismos [54][3]. Esto, al ser un modelo abstracto, puede aplicarse a diferentes escenarios que incluso no parezcan tener relación con el juego estudiado. Desde luego, los juegos de suma–cero no resueltos representan un problema de interés general, porque al no tener una solución específica, se tiene que trabajar con heurísticas y algoritmos que den una mejor aproximación a la solución.

El objetivo principal en la Teoría de Juegos es estudiar el comportamiento racional en situaciones de interacción (lo que se denomina un juego) entre diferentes agentes (jugadores), en donde los resultados están condicionados a las acciones de los mismos [69].

1.2. El problema

Se busca la solución de los juegos de suma–cero e información perfecta, a partir de sus características particulares (lo cual se denominan configuraciones o patrones ventajosos). En pocas palabras se busca la elección de la conducta óptima cuando los beneficios de cada opción no están determinados de antemano sino que dependen de las elecciones de otros individuos.

La búsqueda de patrones ventajosos puede considerarse como una importante alternativa a los esquemas tradicionales como son la matriz de pagos o la búsqueda Minimax [80]. Estas dos últimas alternativas engloban muchos de los juegos "que jugamos" y es claro no son suficientes en muchos casos por la propia naturaleza de los juegos en estudio, es decir, no pueden generar a través de estos mecanismos las soluciones óptimas. Además, hay que reconocer que algunos juegos simplemente no tienen una solución final, es decir, no se pueden enumerar todas las posiciones en donde un jugador obtiene ventaja para ganar o simplemente para no perder.

[2]En el contexto de este trabajo nos estaremos refiriendo siempre a juegos de suma–cero e información perfecta

[3]Michael Kearns establece que *para la ciencia de la computación y del lado de la inteligencia artificial, las motivaciones incluyen el uso de la teoría de juegos como un principio para diseñar algoritmos distribuidos y protocolos para redes, así como para los fundamentos de agentes autónomos involucrados en actividad cooperativa y competencia estratégica.*

1.3. La hipótesis

En muchos juegos de suma–cero e información perfecta existen patrones, configuraciones que se repiten con frecuencia. El principal objetivo de esta investigación es identificar los patrones como ventajosos o ganadores, incluso en etapas tempranas del juego.

Básicamente se busca:

- Reconocer patrones ventajosos o ganadores en los juegos de suma–cero e información perfecta.
- Caracterizar qué definen estos patrones ventajosos.
- Cómo valorar esos patrones ventajosos, posiblemente a partir de una evaluación numérica.
- Cuáles son los elementos comunes que comparten estos patrones.
- La posibilidad de generalizar los patrones como una solución.

Si tomamos el caso del juego del gato, por ejemplo, las configuraciones ganadoras son triviales de definir y tan es así que podemos saber cuando una posición en el tablero del gato es un empate o un triunfo para alguno de los jugadores. Incluso podemos saber esto prácticamente desde los primeros movimientos de los jugadores. Algo similar puede decirse del juego de NIM.

Como caso de estudio tomaremos el juego del ajedrez, en donde las configuraciones de piezas y peones en el tablero se basan en los lugares que ocupan y las casillas que dominan. Hay que buscar una manera de definir estas configuraciones, estos son los *chunks* o patrones y a partir de esto, tratar de demostrar la hipótesis, a través de los criterios dados arriba.

1.4. La aproximación

Para intentar demostrar la hipótesis, se deben investigar los patrones primarios que definen las posiciones ganadoras o visiblemente ventajosas. El tipo de configuraciones que pueden presentarse así como los elementos generadores de las mismas. Se debe buscar entonces establecer qué elementos

las componen. Desde luego, hay que partir de juegos conocidos de suma–cero como el NIM, el Gato, las damas inglesas y el ajedrez, por ejemplo, pero en última instancia se busca generalizar los *resultados* a cualquier juego de suma–cero e información perfecta.

Para ello, para el caso particular del ajedrez, se desarrolla un lenguaje de descripción de patrones, el cual permite definir sin ambigüedad, las configuraciones de las piezas en un tablero de ajedrez, generalizando las acciones de ataques y defensas que pueden realizar[4]. Estos son los patrones propuestos como ganadores o como aquéllos en donde uno de los jugadores tiene una ventaja tangible.

El siguiente paso es formalizar el lenguaje de descripción como un lenguaje algebraico. Un lenguaje de esta naturaleza, con una sintaxis y semántica precisa, se denomina un lenguaje formal. Estos se definen a través de dos conjuntos de reglas [27]:

- **Sintaxis**: reglas precisas que indican qué símbolos son permitidos y cómo pueden ponerse juntos en una expresión legal.
- **Semántica**: reglas precisas que indican el significado de los símbolos y de las expresiones legales.

La descripción algebraica de un lenguaje formal permite estudiar, primero, los aspectos puramente sintácticos de tales lenguajes, es decir, sus patrones internos estructurales: segundo, es la base para definir la gramática de los lenguajes de programación y formalizar versiones de subconjuntos de lenguajes naturales, en donde las palabras del lenguaje representan conceptos que se asocian con significados particulares, es decir, la semántica. En su forma más general, un lenguaje algebraico es el estudio de un conjunto de símbolos y las reglas para manipular dichos símbolos [50].

1.5. La contribución

El enfoque tradicional en el estudio de los juegos de suma–cero e información perfecta, ha sido la creación de árboles de movimientos y jugadas, y –a través de una función de evaluación– definir qué jugador tiene la ventaja,

[4]El Gran Maestro danés Bent Larsen, en su poco conocido libro sobre patrones en las aperturas en ajedrez, indica [56]: *El ajedrez es por naturaleza un juego construido sobre la comunicación –un lenguaje marcado por la agresión– una discusión.*

a partir de los valores que se entreguen en los nodos terminales, esto es, el algoritmo *Minimax* (Capítulo 2 – Algunas definiciones de teoría de juegos).

- Una primera contribución es el poder expresar los patrones de los juegos de suma–cero como patrones, los cuales se basan en gran medida en los trabajos del arquitecto Christopher Alexander [3], y se refieren a soluciones a problemas comunes, que ocurren una y otra vez en el contexto de la arquitectura y que se han pasado al diseño de software. Alexander define un patrón como **una regla de tres partes, la cual expresa una relación entre cierto contexto, un problema y una solución**. (Capítulo 4) [59, 3].

- El análisis realizado en este trabajo muestra que la identificación de olinomioatrones ventajosos es una alternativa posible para hallar una heurística que muestre un comportamiento de buenas jugadas a realizar en los respectivos juegos de suma–cero. La contribución parte de evitar tener que analizar todo el árbol de variantes y valorarlo nodo por nodo y con diversos algoritmos para saber cuál es la mejor jugada. Nuestro enfoque puede orientar al jugador (o incluso a un programa de computadora), a hallar una solución eficiente a los problemas planteados en el juego (Capítulo 5 –Funciones de evaluación *versus* patrones).

- Para el caso del ajedrez, se define un lenguaje de descripción de patrones ventajosos, el cual es aplicado a diversas posiciones en donde se puede observar cómo los patrones existen y se tratan cada uno de ellos de forma similar (Capítulo 3 – Trabajo relacionado). El lenguaje de patrones generaliza particularmente el problema de las configuraciones ventajosas y da una solución general al problema en lo que se refiere a tratar con heurísticas (Capítulo 7 – Descripción del lenguaje de patrones) [60].

- Una contribución más parte del formalizar este lenguaje de patrones para juegos de suma–cero e información perfecta en un álgebra de conjuntos, la cual se muestra como no-conmutativa y que puede representarse fácilmente en los lenguajes de programación, particularmente en aquéllos que se denominan declarativos o funcionales (capítulo 7 – Hacia un álgebra de conjuntos para los patrones en ajedrez).

1.6. Estructura de la tesis

La tesis está estructurada de la siguiente manera:

- **Capítulo 2. Antecedentes.** Se introduce la Teoría de Juegos, la representación de los mismos y de los juegos de suma–cero e información perfecta en particular. Se definen las técnicas más usadas para el tratamiento de esta clase de juegos, como el procedimiento *Minimax*. También se definen los patrones para este tipo de juegos, las formas más comunes de los mismos, la representación gráfica, la clasificación y el lenguaje de patrones.

- **Capítulo 3. Trabajo relacionado.** Este capítulo introduce una revisión de los trabajos relevantes en el área, como puede ser el análisis del juego de NIM, las primeras aproximaciones de patrones al juego del ajedrez y ya propiamente el desarrollo de los últimos años de los *chunks*, configuraciones o patrones, que se han desarrollado particularmente en el ajedrez.

- **Capítulo 4. Patrones de diseño para el caso del ajedrez**. En este capítulo se describe un conjunto de patrones característicos del ajedrez, conocidos desde hace mucho tiempo en la literatura especializada del tema. Se habla de temas como *la fórmula de Tarrasch*, el regalo griego, la regla del cuadrado del peón, la clavada, la distracción el rey ahogado, la jugada intermedia y la destrucción de la estructura de peones.

- **Capítulo 5. Juegos de suma–cero y análisis de de sus patrones ventajosos más significativos.** En este capítulo se definen los patrones más importantes para un conjunto de juegos de suma–cero e información perfecta como son el juego del gato, el NIM y el ajedrez. Se plantea un lenguaje de descripción de patrones de ajedrez que permite representarlos de manera simple. Se definen los elementos de dicho lenguaje y su aplicación en las diferentes posiciones analizadas.

- **Capítulo 6. Funciones de evaluación *versus* patrones**. En general los juegos de suma–cero e información perfecta se resuelven a través de algoritmos como Minimax o Alpha-Beta, considerando una función de evaluación y una estructura arbórea, donde se van desarrollando los nodos de las diferentes jugadas y respuestas que pueden darse a partir

de una posición dada en el juego. Este enfoque no contempla el uso de patrones conocidos como ventajosos, incluso en etapas tempranas del juego, como en el juego del Gato o en el NIM. En este capítulo se analizan ambos enfoques de donde saldrán las correspondientes conclusiones.

- **Capítulo 7. Formalismo del lenguaje de descripción de patrones.** En este capítulo se describe el lenguaje de patrones en una descripción algebraica formal. Se busca desarrollar un modelo algebraico para la descripción de las posiciones de ajedrez. En este marco, los objetos ajedrecísticos consisten en primitivas como las piezas, el tablero de 64 casillas, así como los movimientos (ataques y defensas) de las diferentes piezas, además de las operaciones que transforman los objetos ajedrecísticos en las diferentes posiciones que pueden encontrarse. Se debe tener operaciones que puedan ser combinadas para generar incluso macro-operaciones. Todo esto puede hacerse en el marco del álgebra de conjuntos, de forma que la descripción de las posiciones en ajedrez pueda ser claro y preciso. De esta forma surge una descripción matemática para el juego del ajedrez. En principio este resultado podría expandirse a otros juegos de suma-cero e información perfecta. Finalmente, se muestran los resultados obtenidos.

- **Capítulo 8. Conclusiones.** Aquí se muestran los resultados obtenidos y las conclusiones pertinentes al trabajo realizado, apuntando además, al posible trabajo futuro que puede seguirse desarrollando en el tema que nos ocupa.

Capítulo 2

Antecedentes

En el presente capítulo se definen los elementos necesarios de la teoría de juegos de suma–cero: la definición matemática de un juego, las formas normal y extensiva, las funciones de utilidad que definen el valor del movimiento realizado por cada jugador y el teorema Minimax, así como el llamado "equilibrio de Nash". Por otra parte, se definen los patrones y del cómo esta idea pudiese ayudar a plantear soluciones a problemas comunes, que ocurren una y otra vez en el contexto de la arquitectura y diseño de software.

2.1. Teoría de juegos

Un juego es cualquier problema de decisión donde el resultado del mismo depende de las acciones de uno o más agentes[1]. Se puede definir como una descripción formal de una situación estratégica [78]. El objeto de estudio de la teoría de juegos es precisamente el juego, el cual es un modelo formal de una situación interactiva que en muchas ocasiones es de conflicto. En muchos casos, se involucran muchos jugadores, aunque en un juego con un solo jugador hablamos de un problema de decisión.

Hay muchos tipos de juego, por ejemplo los cooperativos, en donde el juego es una descripción que especifica las ganancias de un grupo potencial de individuos, una coalición, que se puede obtener por la cooperación de sus miembros. Otros son los juegos no cooperativos, en donde se modela explícitamente el proceso en el que los jugadores toman decisiones basadas

[1]Brian Weatherson, Lecture Notes on GAME THEORY, http://brian.weatherson.org/StA-GameTheoryNotes.pdf

en su propio interés. Estos son los que nos interesan en este trabajo.

Un argumento que se asume en general en teoría de juegos es que los jugadores son racionales, es decir: un jugador racional es aquél que siempre elige una acción la cual le da la ventaja que está buscando dado lo que espera del oponente. En un sentido estricto esto implica que cada jugador busca maximizar su utilidad[2].

Otro punto importante, establecido por John von Neumann y Oskar Morgenstern [80], es que demostraron que cualquier juego de suma–cero que involucre a más de dos jugadores es de hecho una forma general de un juego de suma–cero en donde sólo participen dos personas. Esto significa que los juegos de suma–cero para dos jugadores forman el núcleo de la teoría de juegos.

2.1.1. Representación de los juegos

Un juego debe especificarse por una serie de elementos [70, 65, 5]:

- Los jugadores involucrados.
- La información disponible por cada jugador en cada momento.
- Las acciones disponibles por cada jugador en su momento.
- La utilidad o ventaja obtenida por sus acciones.

Podemos expresar esto como: N = $\{1, \ldots, n\}$ como el conjunto de jugadores. $(A_1, ..., A_n)$ son las acciones posibles que dispone cada jugador. $(u_1, \ldots, u_n)$ son las funciones de utilidad que determinan el valor del movimiento realizado (el pago). Cada una de estas funciones está definida en n variables diferentes, una por cada jugador, por ejemplo, un polinomio[3].

Definimos un juego como $G = (N, \{A_i\}\, donde\, 1 \leq i \leq n), \{u_i\}\, donde\, 1 \leq i \leq n$ en donde G es el juego $(game)$ [70, 5].

[2]Shor, Mikhael, "Rationality", http://www.gametheory.net/dictionary/Rationality.html (2005)

[3]Muchas funciones de evaluación o utilidad son lineales, por ejemplo, en ajedrez se considera el material (las piezas en el tablero), como una de las más importantes [75], pero podrían definirse funciones de evaluación o utilidad que sean no lineales

2.1.2. Juegos de suma–cero

Los juegos de suma–cero son un caso particular de la teoría de juegos, que se definen donde la ventaja de un jugador es estrictamente igual a la desventaja del segundo jugador. Podemos definir un juego de la siguiente manera [13, 12]:

Definición 1. La forma estratégica (también llamada **forma normal**), de un juego de suma–cero para dos personas se da por una tripleta (X, Y, A), en donde:

- X es un conjunto no-vacío, el cual es el conjunto de estrategias del jugador I
- Y es un conjunto no-vacío, el cual es el conjunto de estrategias del jugador II
- A es una función valuada en los reales, definida en $X \times Y$. Así entonces, $A(x, y)$ es un número real para todo $x \in X$ y todo $y \in Y$.

En esta definición podemos englobar los juegos con un número finito de combinaciones, como pueden serlo el ajedrez o el juego del gato. Hablamos de estrategias, las cuales se definen como la manera en cómo se conduce el juego, a través de las reglas definidas para el mismo y qué jugada se puede hacer en toda situación que pueda ocurrir. Para algunos juegos de suma–cero, el escribir todas las estrategias posibles puede no ser posible ni práctico, pero se pueden generar heurísticas que instruyan las posibles estrategias a seguir [69]. El conjunto de todas las estrategias posibles del primer jugador se denota como X.

Existen dos tipos de estrategia: *pura* y *mixta*. Los elementos de X o Y definen las estrategias puras. Cuando se decide seguir una estrategia pura al azar en diferentes proporciones, se dice que estamos en una estrategia mixta [61].

Una estrategia mixta puede ser implementada mediante una serie de mecanismos: una heurística, el azar (tirar una moneda o un dado), etcétera. Pero la meta final es obtener la ventaja. Para ello, se usa el siguiente teorema:

Teorema 1. Minimax. En un juego de suma–cero (X, Y, A), se dice que es un juego finito si ambos conjuntos de estrategias X y Y son conjuntos

finitos. Este es el teorema fundamental de la teoría de juegos, debido a von Neumann [80], en donde para todo juego finito de suma–cero, se tiene que:

- Hay un número V, llamado el valor del juego
- Hay una estrategia mixta para el jugador I tal que el promedio de la ganancia de I es al menos V, sin importar lo que haga II, y
- Hay una estrategia mixta para el jugador II, tal que la pérdida promedio de II es al menos V, sin importar lo que haga.

Si $V = 0$ decimos que el juego es parejo o justo; si V es positivo, entonces decimos que el juego favorece al primer jugador (al I), mientras que si V es negativo, entonces decimos que el juego favorece al jugador II [80].

También podemos representar un juego de suma–cero en su **forma estratégica**, (X, Y, A) mediante una matriz de pagos A [13].

Si X=$\{x_1, \ldots, x_m\}$ y Y=$\{y_1, \ldots, y_n\}$ entonces la matriz de pagos es:

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}$$

donde $a_{ij} = A(x_i, y_j)$

Una opción más es representar un juego de suma–cero en forma de **patrones**, los cuales se definen como situaciones con características específicas dentro de un juego y que buscan –al menos– obtener una ventaja en el mismo. Dependiendo del juego que se trate, las estrategias X y Y pueden mostrarse como una secuencia de movimientos o jugadas $\{M_1, \ldots, M_n\}$, las cuales pueden hacer el cálculo de V y así saber quién tiene la ventaja. En este caso las estrategias de cada jugador son en realidad secuencias de movimientos, a los que llamamos patrones.

2.1.3. Juegos de suma-cero y de información perfecta

En los juegos no cooperativos existen aquéllos que son denominados de suma–cero, que son una representación de una situación en donde la utilidad

o ventaja de cada jugador está exactamente balanceada por la pérdida de la utilidad o ventaja de los otros jugadores. Si sumamos las ganancias y las pérdidas totales, la suma debe dar cero. Un juego de esta naturaleza se define como un juego estrictamente competitivo. Es decir:

$$u_1(w) + u_2(w) + u_3(w) + ... + u_n(w) = 0$$

para cada w en el conjunto Ω de resultados finales. Así, $u_1 : \Omega \to \mathbb{R}$ y $u_2 : \Omega \to \mathbb{R}, ..., u_n : \Omega \to R$, son las funciones de utilidad para cada uno de los n jugadores [12].

Algunos de estos juegos tienen en ocasiones una característica conocida como juegos de información perfecta, lo cual ocurre cuando los jugadores tienen toda la información disponible a todo momento. No hay nada escondido. El ajedrez es un juego de información perfecta. El póker, en cambio, no lo es, pues ningún jugador sabe qué cartas tienen los otros jugadores [13, 46].

2.2. Algunas definiciones de teoría de juegos

Un juego es pues una situación en la que dos jugadores seleccionan cursos de acción y en donde el resultado se ve afectado por la combinación de selecciones tomadas en el transcurso del juego. Se tienen entonces siguientes características [80]:

- Hay dos jugadores.
- Hay un conjunto de reglas que especifican cuáles cursos de acción se pueden seleccionar (jugadas) y los jugadores las conocen.
- Hay un conjunto bien definido de estados finales para saber en qué momento termina la competencia.
- Los pagos asociados con cada estado final posible se especifican *a priori* y cada jugador los conoce.
- Cuando cada jugador selecciona su propio curso de acción se dice que ocurre una jugada o un movimiento. Un conjunto de reglas que especifican cuál de las alternativas disponibles debería tomar en cada jugada, es una estrategia para cada jugador dado. La teoría de juegos busca estrategias que maximicen o minimicen alguna función objetivo.

- Una solución a un juego se obtiene cuando se determina la mejor estrategia para cada jugador, y esa mejor estrategia se define en términos de una función objetivo específica.
- La función objetivo apropiada depende de la clase de conocimientos que tengan los jugadores acerca de las alternativas de cada uno de los otros.
- Si cada jugador conoce lo que va a hacer el otro exactamente, se tiene una situación determinística y la función objetivo es maximizar la utilidad. Si se conocen las probabilidades se tiene una situación de incertidumbre que se resuelve con funciones objetivos de tipo *minimax* o *maximin*.
- Según el criterio minimax, un jugador elige la estrategia que minimice la máxima pérdida esperada para sí mismo; según el criterio maximin el jugador busca maximizar el pago esperado mínimo para sí mismo.
- La teoría de juegos se preocupa por el tipo de situación de incertidumbre. De esa manera, su objetivo ha sido convertir el tipo de situación de incertidumbre en un tipo de situación de certeza utilizando ciertas suposiciones racionales con respecto a los jugadores.
- Se supone que cada jugador actúa para maximizar su utilidad esperada, es decir, su beneficio. A partir de esta suposición y de acuerdo a los criterios minimax y maximin, cada jugador actúa para maximizar su ganancia mínima o minimizar su pérdida máxima.

Uno de los objetivos primordiales de la teoría de juegos es establecer criterios que se denominan "racionales" para seleccionar una estrategia, lo cual implican dos suposiciones importantes:

- Ambos jugadores son racionales.
- Ambos jugadores eligen sus estrategias solo para promover su propio bienestar.

Cabe señalar que cuando se habla de juegos estrictamente (como una analogía al conflicto), en términos sociales, muchas veces se decide que al final del juego se hacen *pagos* entre los jugadores de acuerdo con lo determinado

en las reglas del juego mismo. Esto se hace siempre en juegos de azar, por ejemplo. En ocasiones los jugadores llevan registro de lo que han hecho en el juego, una especie de *puntuación*, de manera que al final del mismo se calculen los pagos de acuerdo con los números calculados, como una forma de evaluar la habilidad de cada participante en el juego. Sin embargo, hay ocasiones en donde no hay ningún tipo de puntuación y simplemente se anuncia si alguien ha ganado o perdido, como en el caso del ajedrez o el juego del gato, por ejemplo. En estos casos puede ocurrir que muchas veces no hay pago (monetario, por ejemplo), por ello.

Pero incluso cuando no hay intercambio de dinero (que es un atractivo adicional en muchos juegos), los jugadores muchas veces derivan placer o dolor a partir de las posiciones a las que se llegan en el juego, por lo que bien podría hallarse un equivalente a los pagos que se realizan al terminar los juegos en donde no esté involucrado el dinero necesariamente, pero esto parece caer más en el terreno de la filosofía [61].

De acuerdo con lo anterior, **podemos considerar el "bienestar" en la decisión de un jugador como el beneficio obtenido, es decir, podemos compararlo con el término "ventaja"**.

2.2.1. Representación normal y extensiva

Los juegos no cooperativos se pueden expresar en dos formas, una llamada "extensiva" y otra denominada "normal". La primera se utiliza para formalizar los juegos que tienen secuencias de tiempo en sus movimientos. Los juegos de esta naturaleza pueden expresarse en forma de árboles en donde cada nodo representa un punto para que un jugador tome una decisión. A este tipo de representación se le llama un "árbol de decisión" [42].

La forma "normal", en cambio, se representa en general por una matriz de que muestra a los jugadores, las estrategias y las utilidades. En su forma más general, pueden ser representadas por cualquier función que asocie la ventaja de cada jugador mencionando todas las posibilidades de las combinaciones de acciones. Esto es un modelo exhaustivo que funciona para muchos juegos no cooperativos pero desde luego, no para todos. En un juego representado en su forma normal, cada jugador tiene listadas todas las posibilidades de acción, por lo que no necesita de hecho de saber las acciones de los otros jugadores. Si los jugadores tienen información del rival en algún momento, es preferible usar la forma extensiva [5].

La forma normal (también llamada "forma estratégica") de un juego es

una matriz de pagos, que muestra los jugadores, las estrategias, y las recompensas en cada caso [42]. Hay dos tipos de jugadores; uno selecciona la fila y otro la columna. Cada jugador tiene dos posibles estrategias, que están especificadas por el número de filas y el número de columnas. El primer número es la recompensa recibida por el jugador de las filas (A) y el segundo es la recompensa del jugador de las columnas (B).

	jugador B selecciona izquierda	jugador B selecciona derecha
jugador A elige arriba	4,3	-1, -1
jugador A elige abajo	0,0	3, 4

Figura 2.1: Representación normal

La matriz (ver figura 2.1) es una representación en la forma normal de un juego en el que dos jugadores hacen sus movimientos en forma simultánea, lo cual quiere decir que no conocen el movimiento del otro jugador. Cada jugador recibe las recompensas tal y como se especifica de acuerdo a la combinación realizada, a las opciones elegidas. Por ejemplo, si el jugador A selecciona arriba y el jugador B selecciona izquierda, el jugador A recibe 4 y el jugador B recibe 3. En cada celda o casillero, el primer número representa la recompensa del jugador de las filas (en este caso el jugador A), y el segundo número representa la recompensa del jugador de las columnas (en este caso el segundo jugador, el B). El análisis por matriz de ganancias puede ser aplicado cuando hay un conjunto finito de decisiones discretas alternativas.

La representación en forma extensiva, por su parte, modela juegos con algún orden que se debe considerar. Los juegos se presentan como árboles. Cada nodo representa un punto donde el jugador toma una decisión. El jugador se especifica por un número situado junto al vértice. Las líneas que parten del nodo representan acciones posibles para cada jugador. Las recompensas se especifican en las hojas del árbol.

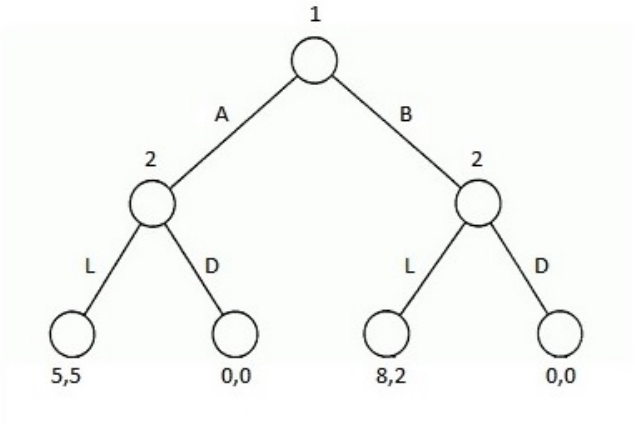

Figura 2.2: Representación extensiva

Considérese la figura 2.2, que muestra el juego entre dos jugadores, 1 y 2: El jugador 1 mueve primero y selecciona entre A o B. El jugador 2 ve el movimiento del jugador 1 y selecciona L o D. Si el jugador 1 se decide ir por B y entonces el jugador 2 selecciona L, entonces el jugador 1 obtiene 8 y el jugador 2 obtiene 2.

2.2.2. Equilibrio de Nash

El equilibrio de Nash, también llamado "equilibrio del miedo" es un "concepto de solución" para juegos con dos o más jugadores[4], el cual asume que:

- Cada jugador conoce y ha adoptado su mejor estrategia, y
- todos conocen las estrategias de los otros.

Consecuentemente, cada jugador individual no gana nada modificando su estrategia mientras los otros mantengan las suyas. Así, cada jugador está ejecutando el mejor "movimiento" posible teniendo en cuenta los movimientos de los demás jugadores [80].
El concepto de equilibro de Nash se da en muchos juegos. Sea un juego $G = \{S_1, ..., S_n; u_1, ..., u_n\}$, con estrategias $\{s_1^*, ..., s_n^*\}$, las cuales forman un

[4]Un concepto de solución es en realidad una regla formal que predice las estrategias que los participantes buscan tener a fin de lograr los mejores resultados

equilibrio de Nash en estrategias puras si, para cualquier i, $s_i^* \in S_i$ es la mejor respuesta a las estrategias $\{s_1^*, ..., s_i - 1^*; s_i^*, s_i + 1^*, ...s_n^*\}$ de los otros $n-1$ participantes (o jugadores en el juego). Es decir, $u_i(s_1^*, ..., s_i - 1^*; s_i^*, s_i + 1^*, ...s_n^*) \geq u_i(s_1^*, ..., s_i - 1^*; s_i^*, s_i + 1^*, ...s_n^*)$ para cualquier jugador i y para cualquier $s_i \in S_i$ [13].

Esto quiere decir que todos y cada uno de los jugadores resuelven individualmente el problema, alcanzándose el equilibrio de Nash cuando todos, de forma simultánea, obtienen su máximo. Así entonces, ningún jugador tiene incentivo alguno para modificar individualmente su estrategia.

Dicho de otra manera, las estrategias de cada jugador deben ser la mejor respuesta a las estrategias de los otros jugadores. Todos los jugadores buscan elegir estrategias de equilibrio y a ninguno le conviene desviarse de su mejor estrategia. En caso de alguna desviación, entonces no estamos en el equilibrio de Nash [12, 13].

2.2.3. Decisiones basadas en Minimax

Un modelo de decisión muy usado es el teorema Minimax, en el cual se intenta que un jugador minimice las pérdidas en su peor escenario (pérdida máxima). Originalmente, se formula para los juegos de suma–cero entre dos jugadores, cubriendo el caso en donde los participantes hacen movimientos de forma alternada.

El teorema establece que en los juegos de dos personas de suma–cero, donde cada jugador conoce de antemano la estrategia de su rival así como los resultados de la misma, existe una estrategia que permite a ambos jugadores minimizar la pérdida máxima esperada. Esto quiere decir que en particular cuando se examina cada estrategia disponible, un jugador debe tomar en cuenta todas las respuestas posibles del rival y la pérdida máxima que puede acarrear cada una de ellas. El jugador juega, entonces, con la estrategia que resulta en la minimización de su máxima pérdida. Dicho de manera precisa [65]:

Para todo juego de dos personas, de suma-cero, con un número finito de estrategias, existe un valor V y una estrategia para cada jugador de tal manera que:

1. Dada la estrategia del segundo jugador, el mejor resultado para el jugador 1 es V, y

2. Dada la estrategia del primer jugador, el mejor resultado para el jugador 2 es -V.

A manera de algoritmo, Minimax puede resumirse en los siguientes pasos [72]:

1. Generación del árbol de juego. Se generan todos los posibles nodos hasta llegar a un estado terminal.

2. Cálculo de la función de evaluación para cada nodo terminal.

3. Calculo del valor de los nodos superiores a partir del valor de los inferiores. Podremos decidir, de acuerdo al nivel en donde nos encontremos, si es MAX o MIN se eligen los valores mínimos y máximos representando los movimientos del jugador y del oponente, (de ahí el nombre de Minimax).

4. Seleccionar la jugada tomando en cuenta los valores que han llegado al nivel superior.

Cabe decir que una función de evaluación es aquélla que se usa en los juegos para estimar el valor o la bondad de una posición. También se le conoce como función de evaluación heurística. Ésta se aplica a cada nodo en el árbol del juego [12].

2.2.4. Ejemplo simple de Minimax

Tomemos un juego imaginario en donde los valores de la función de evaluación son de 1 a 9. En los movimientos propios, se busca maximizar la utilidad, mientras que en el turno del rival, se busca minimizar la utilidad o ventaja del contrario (véase la figura 2.3). El primer paso consiste en calcular todos los valores de la función para cada nodo terminal (los que están en círculo). Acto seguido calcularemos (en el nivel 4), los valores que del nivel 5 son mínimos. Así hallamos 5, 2 y 1. Subimos al nivel anterior (el 3) y maximizamos la utilidad, que en este caso es 5 y 9. En el segundo nivel tenemos entonces el turno de minimizar (MIN), que nos da 5, 3 y 1. Finalmente llegamos al nivel 1 en donde se selecciona el valor que maximice (MAX) nuestra ventaja, 5.

La función de evaluación puede indicarnos con certeza el valor final calculado de cada posición. Esto puede complicarse en la medida que el sistema

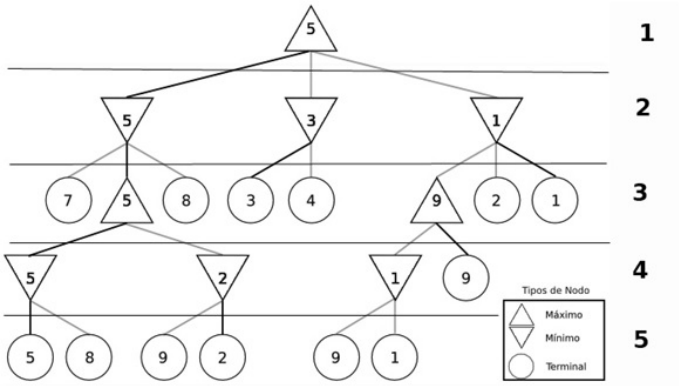

Figura 2.3: Árbol Minimax de un juego imaginario

estudiado es más complejo. Por ejemplo, una función de evaluación para el juego del gato es muy sencilla, mientras que para el ajedrez, la función de evaluación puede ser mucho más compleja [12] y aún así, no se garantiza que sea la mejor posible.

2.2.5. Un modelo formal de los juegos de suma–cero

Considérese un juego de suma–cero que se juega en un espacio de estados finitos, en donde cada jugador tiene un número finito de posibles acciones que puede tomar. Definimos formalmente este escenario como:

$$G = (S, P, (A_i, a_i, U_i)_{1 \leq i \leq |p|}, Q)$$

En donde tenemos los siguientes parámetros:

- S se define como $S = s_1, s_2, ..., s_n$, un conjunto finito de estados.
- P se define como $p = \{p_k\}_{k=1,2}$, un conjunto de jugadores.
- A_i en donde $\forall(p_k \in p)\exists A_i = a_1, a_2, ..., a_n$. Esto es, para cada jugador p_k existe un conjunto finito de acciones para ese jugador.
- a_i en donde $a : S \rightarrow A_i; i = 1, 2$, un mapeo que asigna a cada estado $s \in S$ el conjunto de acciones $a_i(s)$ que están disponibles al jugador i en el estado S.

- $Sa = (s, a) : s \in S, a = (a_i), a_i \in a_i(s); 1 \leq i \leq |p|$, el conjunto de todas las acciones para cada jugador.
- $U_i : Sa \to R, i = 1, 2$, para cada jugador U_1 asigna un pago al jugador p_1 cuando la acción correspondiente es jugada.
- Q en donde $Q : SA \to P(S)$, on Q la distribución de la probabilidad sobre S.

La magnitud de la recompensa en un juego de suma–cero entre dos personas es la recompensa para el atacante, contra la pérdida del defensor. Esto además tiene que cumplir con la propiedad de suma–cero, en donde $X + (-X) = 0$ [52].

Estamos ante un juego en donde los jugadores tratan de maximizar ganancias o minimizar pérdidas. En este caso el modelo es de atacante y defensor. Podemos representar a los jugadores del juego como p_1, el defensor y p_2 el atacante. El espacio de acciones de los jugadores es el conjunto de posibles ataques y movimientos de defensa respectivamente. El modelo encapsula cada ataque o defensa como una sola acción que logra una meta específica [83].

Por lo tanto, el espacio finito de acciones para ambos jugadores, el defensor p_1 y el atacante p_2 se definen como:

$$A_1 = P_1^a = a_1, a_2, ..., a_n$$

$$A_2 = P_2^a = a_1, a_2, ..., a_n$$

Un juego de suma–cero entre dos jugadores frecuentemente transita de un estado a otro de acuerdo con una función de evaluación que incluso puede ser probabilística. La probabilidad del estado de transición es una función de las acciones de ambos jugadores y del estado actual en el juego. Estas probabilidades no sólo determinan los estados sino que se incorporan a la solución del juego, en donde se mezclan muchas veces las estrategias de los jugadores [52].

Un juego G consiste entonces en un conjunto finito de estados (o posiciones):

$$S = \{s_1, s_2, ..., s_n\}_{1 \leq n \leq |s|}$$

que representan la situación en el juego en un momento dado.

En los juegos de "atacantes y defensores", las acciones se desarrollan observando las acciones de los defensores para proteger un recurso, un pedazo de territorio, etcétera, mientras que un número de atacantes intentan destruir o capturar el mismo recurso o territorio que está siendo defendido. En algunos juegos las ganancias y pérdidas de cada jugador puede expresarse como matrices de pago o bien como árboles de decisión en donde se aplica el criterio Minimax[5].

Asociado con cada estado s_k hay una matriz del juego G^k. La transición del estado s_k a otro s_l depende del resultado de G^k.

2.3. Patrones de diseño

En ingeniería de software se habla de patrones, los cuales se basan en gran medida en los trabajos del arquitecto Christopher Alexander [3], y se refieren a soluciones a problemas comunes, que ocurren una y otra vez en el contexto de la arquitectura y que se han pasado al diseño de software. Alexander define un patrón como **una regla de tres partes, la cual expresa una relación entre cierto contexto, un problema y una solución** [3].

Cada patrón no es un diseño finalizado que puede ser transformado directamente en código fuente. Es de alguna manera una descripción, una receta acerca de cómo resolver un problema que se repite en diferentes situaciones. Para Alexander, cada elemento en el mundo, cada patrón, es una relación entre cierto contexto, un sistema de fuerzas que ocurren repetidamente en dicho contexto y ciertas configuraciones espaciales que permiten que se resuelvan las fuerzas involucradas [19]. El término fuerza se refiere a cualquier aspecto del problema que debe considerarse en la solución.

La definición de un patrón es amplia. Por ejemplo, como elemento de un lenguaje, un patrón es una instrucción que muestra cómo una configuración espacial puede ser usada una y otra vez para resolver un sistema de fuerzas donde hay un contexto que es relevante. Un patrón ataca un problema de diseño recurrente que nace de una situación de diseño específico y presenta entonces una solución.

[5]https://imowensims.wordpress.com/2015/12/29/attacker-defender-games-an-introduction/

2.3.1. Formas de los patrones

Los patrones son como las formas literarias: sonetos, cuentos o novelas, por ejemplo. La forma de un patrón pretende introducir el problema, describir el contexto, es decir, cuándo puede aparecer el problema en cuestión y finalmente, hallar una posible solución. La definición básica es pues que una solución a un problema en un contexto evoca elementos de la forma. Existen muchos componentes de las formas de patrones ya establecidas. He aquí algunos de ellos:

Nombre - Cómo se llama el patrón es importante, porque representa el significado del mismo de manera clara. La elección de un buen nombre puede ayudar a establecer para qué es útil en la comunicación entre desarrolladores. Hay nombres de patrones que describen el contexto para una solución, por ejemplo *A Place to Wait* [38] o *Identify the Nouns* [31]. También pueden describir el problema como en el caso de *Chicken and Egg* [6] o *Developing in Pairs* [25].

Resumen - Es un enunciado que brevemente describe lo que hace el patrón describiendo el problema de diseño que resuelve. Hay patrones que no contienen resumen o tienen nombres alternativos para el mismo. Depende de la forma que se tomen para definir los patrones.

Contexto - En qué situación ocurre un problema es lo que se denomina el contexto. Especifica los elementos de la situación, el tamaño, entorno, lenguaje de programación, etcétera. Se trata de todo lo que define la situación y que si cambia, invalidaría el patrón.

Problema - Describe qué se quiere resolver. En algunas formas de patrones el problema se reduce a una sola pregunta o una formulación breve de la dificultad. Otras formas lo plantean como un ensayo breve que ilustra la necesidad.

Fuerzas - Una definición alternativa de las fuerzas podrían ser los "pros" y "contras" existentes en el problema a resolver. Esto se entiende además como un balance de fuerzas. De acuerdo a Alexander [2], el entendimiento de las fuerzas es crucial para comprender los sistemas. Las fuerzas de hecho ayudan a entender cómo aplicar un patrón efectivamente.

Solución - Esto es lo que se trata de resolver en el enunciado del problema. Algunos patrones no dan una solución completa, sino solamente una parte.

Representación gráfica - Se afirma que un patrón puede ser dibujado, es decir, se puede hacer un esquema gráfico del mismo y Alexander considera esto como la esencia del patrón [2]. De hecho afirma *Si no se puede dibujar un diagrama de ello, no es un patrón* [3]. Cualquier cosa que se piense pueda ayudar al diseñador a entender la relación entre las partes es lo que se puede dibujar. Una representación gráfica de un patrón es una manera gráfica esquemática más que un dibujo.

Consecuencias - Esto es el contexto resultante, que es parecido a una conclusión del patrón. Las consecuencias comunican qué fuerzas han sido resueltas o balanceadas, qué nuevos problemas pueden ocurrir a partir de la aplicación del patrón, qué patrones relacionados pueden seguir. Un patrón no resuelve un problema necesariamente, sino que puede ser la introducción a otros patrones. Los contextos en este caso sirven como relaciones o ligas entre los patrones dentro de un sistema o lenguaje de patrones.

2.3.2. Formas comunes de los patrones

Existen una serie de formas de patrones que son populares, empezando por la *Forma Alexander* [3], basada en el trabajo de Christopher Alexander. Esta debería ser considerada la forma original de los patrones. Las secciones en esta forma no están delimitadas fuertemente y Alexander usa la estructura sintáctica "por lo tanto" (*therefore*), que precede a la solución. Hay también una descripción precisa y clara del problema, una discusión de las fuerzas involucradas, así como la solución.

Otra es la forma GoF *(Gang of Four)* [43], el cual se compone de las siguientes secciones:

Nombre del patrón y clasificación - Es decir, un nombre único descriptivo que ayude a identificar y referirse a un patrón.

Intención - Una descripción de la meta detrás del patrón y la razón para usarlo. A esto también se le conoce como otro nombre para el patrón.

Motivación (Fuerzas) - Un escenario consistente en un problema y un contexto en donde el patrón puede ser usado.

Aplicabilidad - Situación en donde el patrón puede ser usado, el contexto del patrón.

Estructura - Una representación gráfica del patrón. Por ejemplo, un diagrama de clases y de interacción.

Participantes - Una lista de clases y objetos usados en el patrón así como su rol en el diseño.

Colaboración - Una descripción de cómo las clases y los objetos usados en el patrón interactúan unos con otros.

Consecuencias - Una descripción de los resultados, efectos colaterales y problemas causados por el uso del patrón.

Implementación - Es la parte de la solución del patrón.

Código ejemplo - Una ilustración de cómo el patrón puede ser usado en un lenguaje de programación.

Usos conocidos - Ejemplos de usos reales del patrón.

Patrones relacionados - Otros patrones que tienen alguna relación con el patrón que nos ocupa. Discusiones sobre las diferencias y similitudes de los mismos.

La forma GoF se utiliza para diseños de software orientado a objetos.

Otra forma, parecida a GoF es la llamada POSA (*Pattern-Oriented Software Architecture*) [30], que contiene las siguientes secciones:

Nombre - Nombre o resumen corto del patrón.

También conocido como - Estos son otros nombres para el patrón, un alias o si el mismo patrón tiene más de un nombre.

Ejemplo - Es un ejemplo del mundo real en que se demuestre la existencia de un problema y la necesidad de un patrón.

Contexto - La situación en la cual el patrón se aplica.

Problema - El problema que resuelve el patrón, incluyendo una discusión sobre las fuerzas asociadas al mismo.

Solución - El principio fundamental de la solución que sirve como base al patrón.

Estructura - Una especificación de los aspectos estructurales del patrón, incluyendo un diagrama de clases en OMT [71].

Dinámica - Escenarios típicos que describen el comportamiento en tiempos de ejecución del patrón. Los escenarios se ilustran con diagramas de secuencias de mensajes entre objetos.

Implementación - Guías para la implementación del patrón.

Ejemplo resuelto - Discusión sobre aspectos importantes para resolver el ejemplo (no cubiertas en las secciones Solución, Estructura, Dinámica e Implementación.

Variantes - Una descripción breve de variantes o especializaciones del patrón.

Usos conocidos - Ejemplos de uso del patrón a partir de sistemas ya existentes.

Consecuencias - Beneficios y desventajas del patrón.

Finalmente la *Forma Coplien* es similar a la forma Alexander. Define las secciones con títulos:

El nombre del patrón - Se usan sustantivos para los nombres de los patrones. También admite frases verbales.

El problema - Se enuncia frecuentemente como una pregunta o como un reto de diseño.

El contexto - Es la descripción del contexto en donde el problema ocurre y cómo se aplica la solución.

Las fuerzas - Describen los pros y contras del patrón de diseño.

La solución - ¿Cómo se soluciona el problema? Una representación gráfica, un esquema simple (*sketch*) acompaña la solución.

Un razonamiento - ¿Por qué funciona el patrón? ¿Cuál es la historia detrás de él? Esto se extrae del resto de la descripción, con lo cual no se sobrecarga la solución. Se considera al razonamiento como una fuente de aprendizaje más que una fuente de acción.

Contexto resultante - Describe qué fuerzas resuelven el patrón y cuáles se mantienen sin solución. Apunta en muchas ocasiones a otros patrones que podrían ser considerados.

Los mejores patrones son aquéllos que resuelven un problema específico, no solamente es una serie de principios abstractos o estratégicos. Son un concepto probado y no una teoría o especulación. Muestran una solución a un problema no trivial y describen los mecanismos y estructuras de un sistema, no solamente son módulos. Finalmente, los patrones tienen un componente humano significativo. Los mejores patrones explícitamente apelan a la utilidad y a la estética.

2.3.3. La representación gráfica de los patrones

Alexander [2] sostiene la necesidad de una representación gráfica (un *sketch*), el cual es la esencia del patrón mismo y afirma contundentemente que "si no se puede dibujar un diagrama de un patrón, entonces no lo es" [3]. Muchos diagramas se dibujan para ayudar a entender la relación que hay entre las partes. La representación gráfica tiene muchas veces esta estructura. Alexander, como arquitecto, entiende que el propósito de un dibujo arquitectónico es ilustrar la relación entre las partes y no se deben buscar formas realistas y afirma: "Es esencial, por lo tanto, que el constructor construya sólo a partir de dibujos a grandes rasgos, y que realicen los patrones detallados a partir de los dibujos de acuerdo a los procesos dados en el lenguaje de patrones en su mente" [3].

2.3.4. Lenguajes de patrones

Se denomina un lenguaje de patrones a una colección de patrones que se complementan entre sí para generar un sistema [19]. Se trata de una descripción de una arquitectura, un diseño, un entorno (*framework*) o cualquier otra estructura del software. Un ejemplo es el lenguaje de patrones [CHECKS] [28], que genera una arquitectura de manejo de errores para las interfaces hombre-máquina que se ajusten a un contexto específico.

Cabe señalar que el término "lenguaje de patrones" puede confundir y hay autores que consideran mejor llamarlos "sistemas de patrones" [19]. Los lenguajes de patrones están relacionados con la noción de "familia de software" [66], la cual comprende varios miembros relacionados entre sí por lo que tienen en común y por lo que varía entre ellos. Es decir, hablamos de cosas comunes y variables. Así pues, consideramos a un lenguaje de patrones como una colección de reglas para construir a todos los miembros de una familia de software (y sólo miembros de esa familia). Un lenguaje o sistema de patrones no es sólo un árbol de decisión de patrones pues no es una jerarquía. Es decir, el número de caminos distintos a través de un lenguaje de patrones es muy grande.

Los lenguajes de patrones colocan a cada patrón en particular en contexto. Alexander dice [3]: "Cada patrón entonces depende tanto de los patrones más pequeños que contiene, como en los patrones más grandes entre los que se contiene...". Para Alexander, esta red, las ligas o enlaces entre los patrones son tanto casi una parte del lenguaje como los patrones mismos. Esta estructura de la red es la que le da sentido a los patrones individuales pues los ancla y ayuda a complementarlos.

Los patrones definidos permiten, cada uno, realizar a hacer algo en específico. Pero si se piensa en un diseño completo, se requiere que la aplicación esté balanceada a partir de todos estos patrones en conjunto. Un lenguaje –como un todo– genera una familia de arquitecturas y en términos de software, definen un género, pero no en término de protocolos o requerimientos, sino en términos de estructuras básicas y de mecanismos que surgen de la organización [10].

2.3.5. Clasificación de patrones

Los patrones de diseño no tienen todos la misma jerarquía. Jerarquizar es una forma de estructurar y abstraer elementos conceptuales. En programación orientada a objetos tenemos por ejemplo la herencia entre los objetos definidos y esto nos da una jerarquía. Los modelos de categorización de patrones consideran tres niveles o capas de abstracción, los modismos (*Idioms*), que están en la base, los Patrones de diseño (*Design Patterns*) en el medio y Patrones Arquitectónicos (*Architectural Patterns*) en la parte más alta.

- Los modismos son patrones considerados de bajo nivel que dependen de la implementación específica, como por ejemplo, en un lenguaje de

programación [24].

- Los Patrones de diseño están por encima de los modismos. Hay una independencia del lenguaje de programación y en general de la implementación por lo que se consideran prácticas generales para las clases comunes de problemas de software. Gamma *et al* [43] capturan por ejemplo las prácticas del diseño orientado a objetos, el cual hacen independiente de un lenguaje de programación.

- Los patrones arquitectónicos son patrones a nivel de sistemas de software, describe la estructura y funcionamiento básico de un sistema de software como un todo.

Cabe señalar que clasificar los patrones siempre se hace de forma arbitraria y esto puede ser una dificultad. Muchos patrones trascienden los tres niveles mencionados. Por ejemplo, el patrón MVC es, por una parte, un elemento de diseño orientado a objetos para las interfaces con el usuario. Buschmann *et al* [19] lo presentan como un patrón arquitectónico porque representa el principio primario de estructuración al nivel más alto de varios sistemas. Pero los patrones tan amplios suelen ser difíciles de categorizar con la jerarquía que hemos definido.

2.4. Resumen

En este capítulo se plantea los elementos básicos de la Teoría de juegos, el cómo se representan, los diferentes tipos de juego y sus definiciones, así como el concepto de Minimax. Una vez con estos elementos se plantean los patrones de diseño como una manera de representar las situaciones más comunes que pueden ocurrir en una diversidad de dominios, inclusive los juegos. Se definen las diferentes maneras de describir los patrones de diseño y se contrastan sus pros y contras.

Capítulo 3

Trabajo relacionado

En este capítulo se analizan algunos de los artículos más relevantes sobre los juegos de suma–cero e información perfecta. Se hace una revisión sobre el tema de *chunking*, que parece ser una de las direcciones más prometedoras de acuerdo a los trabajos relacionados. Igualmente dos temas más se tratan en este capítulo: Por una parte, la generación de un lenguaje de descripción orientado a los juegos de suma–cero, en particular el ajedrez (el caso de estudio), el cual permita definir conocimiento ajedrecístico de manera fácil y sencilla, para incluso ser ocupado por los motores de ajedrez existente; por otra parte, el problema de la similitud entre las configuraciones que se consideren candidatas a ser ganadoras o al menos ventajosas, o que cumplan con un mínimo de requisitos.

3.1. Zobrist y Carlson: una primera aproximación a los patrones

Albert Zobrist y Frederic Carlson [84] son probablemente los primeros investigadores que intentaron un nuevo enfoque a la creación de un programa de ajedrez. De hecho, en términos generales, todos los programas anteriores a éste se habían escrito siguiendo los lineamientos que Claude Shannon esboza en su famoso artículo para escribir un programa que pudiese jugar al ajedrez [75].

La idea de Zobrist y Carlson se basa en el reconocimiento estructural de patrones y lo aplica al juego de ajedrez. La estructura del juego puede considerarse de enorme complejidad y sin embargo, los seres humanos se saben

guiar en este sistema con bastante éxito. Aparentemente los seres humanos reconocen posiciones similares y se familiarizan con éstas. Así pues, el ajedrez parece ser un caso ideal en el reconocimiento de patrones estructurales.

Zobrist y Carlson plantearon la posibilidad de comunicar el conocimiento del ajedrecista en términos de un lenguaje, el cual debería tener cuatro requerimientos básicos:

- El lenguaje debe ser lo suficientemente poderoso para la descripción de las estrategias ajedrecísticas.
- Debe ser un lenguaje lo suficientemente simple o sencillo para que los ajedrecistas puedan entenderlo a pesar de no tener conocimientos de programación.
- El lenguaje debe poder dar consejos sobre las jugadas que hay que hacer
- El lenguaje debe poderse implementar en una computadora.

Los investigadores trabajan con una serie de primitivas y su sintaxis, es decir, las palabras del lenguaje y su gramática. Entre las primitivas se encuentran: las piezas del ajedrez (12 diferentes piezas, seis blancas y seis negras), las 64 casillas del tablero, el número de movimientos realizados, el número de peones que tiene cada jugador en todo momento así como saber la posibilidad de enrocar de ambos rivales.

La sintaxis del lenguaje permite describir una combinación de las primitivas. Zobrist y Carlson definen el término "patrón" como las instrucciones que describen la posición general de las piezas, que ocurren frecuentemente en el juego de ajedrez entre seres humanos. A una configuración particular de esto se le denomina "instancia". Implementan los investigadores entonces una rutina que puede buscar en una posición todas las instancias de los patrones y grabarlos internamente. A esto le llaman *snapshot* o instantánea.

La rutina que usan los investigadores se define en estos términos:

- Un vector con 50 valores que contienen enteros, entre los que están el número de jugadas, la jugada actual, si alguno de los jugadores se puede enrocar (corto o largo), la suma del valor total de las piezas propias, la suma total de las piezas del oponente, la cantidad de peones del primer jugador, la cantidad de peones del segundo jugador, etcétera.

- 28 arreglos de 8x8 enteros, en donde cada arreglo cumple con una función diferente, por ejemplo, el número de piezas blancas que atacan una casilla, el valor inverso de una pieza (el peón vale en este caso más que las demás piezas), la posición de las casillas débiles (agujeros), dejados en la conformación de peones, la estructura de peones, etcétera.

Un patrón puede ser entonces aquél que detecta todas las instancias en que las piezas blancas atacan a las piezas negras (o que pueden moverse para atacar a las piezas negras). En este caso el propio patrón hace uno o dos movimientos, una especie de *lookahead*.

Zobrist y Carlson definen 45 patrones que reconocen hasta 3000 instancias para una posición dada y cada instancia tiene su propio peso. Por ejemplo, una pieza –una torre– que ataca una dama, tiene más peso que una dama que ataca a una torre. La diferencia entre el valor de las piezas es importante aquí .

El procedimiento se hace en tres pasos:

1. Definir la posición
2. Aplicar patrones
3. *Lookahead*, evaluación y minimax

Cabe señalar que el tercer elemento es exactamente lo que hacen los programas de ajedrez tradicionales. Sin embargo, la diferencia contra las técnicas usuales son los primeros dos pasos.

La idea de Zobrist y Carlson es de los años setenta del siglo pasado y evidentemente la capacidad de los equipos de cómputo era mucho más limitada de lo que podemos tener ahora. La investigación no ha continuado a pesar de su potencial y para entonces se empezaron a desarrollar enfoques partiendo de las ideas de Shannon básicamente. Cabe decir que a pesar de parecer una idea razonablemente buena, los investigadores no han continuado en esta línea de trabajo por razones desconocidas.

3.2. Recuperación de posiciones similares en ajedrez

Este trabajo de Ganguly, Debasis y Leveling, Gareth J.F. y de Johannes y Jones, [44] contiene algunas ideas interesantes para definir una métrica

de similitud de patrones en el juego de ajedrez. Los autores investigan este problema desde el punto de vista de la recuperación de información (*IR – information retrieval*). La idea de hacer este tipo de búsquedas tiene que ver con aplicaciones prácticas, como mostrar las partidas de ajedrez donde un jugador tiene una posición que es similar –de alguna manera– con otras que se han dado y así poder orientar sobre qué jugadas/planes el jugador en cuestión puede realizar. La idea no es nueva en términos de búsqueda, pero el enfoque parece ser al menos diferente.

Los autores plantean otros enfoques que ya se han dado. Por ejemplo, hay buscadores de posiciones que estrictamente analizan configuraciones que se han presentado en una colección de partidas a ver en cuáles se produce una posición específica de las piezas en una configuración específica. A esto le llaman "consulta por ejemplo" (*Query By Example –QBE*), y es una manera elemental de buscar en posiciones. Este enfoque puede hallarse en muchísimas bases de datos de partidas de ajedrez, tanto comerciales como gratuitas (y/o de código abierto). Esto quiere decir que un sistema de esta naturaleza no puede hallar posiciones similares.

Otro enfoque es el que utiliza el software *CQL – Chess Query Language* [26], el cual tiende a ser más flexible y de alguna manera busca hallar la similitud en ciertas posiciones. Sin embargo plantea algunos inconvenientes:

- Debe formularse cada consulta en un lenguaje de consultas que tiene un grado de complejidad.

- Para tener más flexibilidad, CQL hace una búsqueda basada en construir una posición a partir de la consulta para cada partida en la colección de las mismas y reporta los hallazgos. Esta implementación es interesante pero en muchos sentidos muy lenta. Además, tiene la dificultad de que no puede regresar un valor de la similitud de las posiciones, porque solamente usa una variable booleana para reportar los resultados [44].

Así pues, el artículo plantea las siguientes preguntas:

- ¿Cómo puede una posición de ajedrez codificar sus elementos constituyentes como una cadena de símbolos, la cual pudiese ser indexada en un sistema estándar del tipo IR?

- ¿Cómo puede la relación estructural entre las piezas de ajedrez ser representadas en un índice y cómo puede hacerse una función de similitud usando este tipo de búsquedas?

El enfoque mencionado plantea tratar las posiciones de forma parecida cuando se hacen búsquedas sobre una colección de fotografías, en donde se describen de alguna manera los elementos que conforman el rostro[1] [82] y se hace una búsqueda textual basada en las cadenas de símbolos que representan precisamente estas características.

En el mismo caso, los autores del artículo buscan una analogía entre este tipo de búsquedas de rostros y posiciones de ajedrez, en donde hay elementos que plantean una similitud, tanto en las posiciones como en los rasgos característicos de las personas. Hay trabajos que plantean esta posibilidad ya para las posiciones de ajedrez [79].

Sin embargo, el artículo en cuestión define una función de similitud basándose en diferentes factores, pero tomando en cuenta cuál es la posición que tiene la mejor puntuación dada una función de evaluación (que normalmente es un polinomio lineal), a partir del artículo de Claude Shannon [75].

Los factores que tienen que ver con la similitud de posiciones se basan en:

- Casillas que atacan, a donde cada pieza puede llegar
- Conectividad entre las piezas
- casillas atacadas
- casillas defendidas
- Casillas atacadas en rayos X

Los autores hacen, para cada posición, un análisis y conteo de las jugadas que tiene cada pieza en el tablero, qué casillas ataca/defiende, qué piezas ataca/defiende, y le asigna una evaluación a cada una de estos movimientos que tiene cada pieza. Esto es en un afán de poder definir la mejor posición de todas las que encuentre similares.

Como la idea es hacer búsquedas sobre cadena de símbolos, los autores definen algunos, por ejemplo, “>” (ataque), “<” (defensa), “=” (ataque por rayos X).

[1] http://kuznech.com/products/facedetection/

Esto, en la siguiente posición (figura 3.1), se puede describir así:[2]

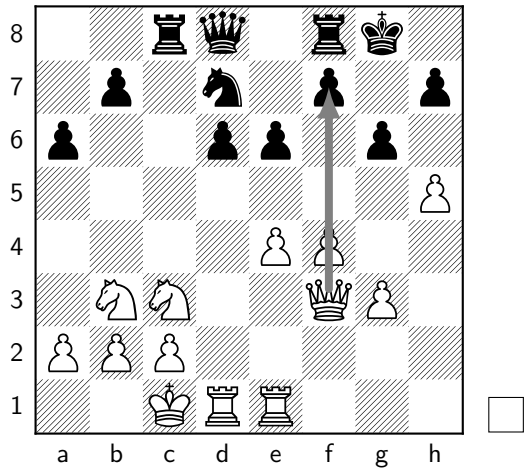

Figura 3.1: **Rayos X de la dama sobre la casilla *f7***

```
rc8 qd8 rf8 kg8 ...
Qf2|0.89 Qf1|0.78 Qg2|0.89 Qh1|0.78 Qe2|0.89 Qe3|0.89 ...
b>Nc3 r>Nc3 p>Ph5 R>pd6 ...
Q<Pe4 Q<Pf4 Q<Pg3 Q<Nc3 Q<Rd1 ...
R=qd8 R=nd7 R=pd6 r=Qf3 r=Pf4 ...
```

Figura 3.2: Rayos X

Cabe señalar que la inclusión de una función de evaluación parece redundante. La realidad es que posiciones similares tienen jugadas similares. Se comprende el esfuerzo por dotar de una manera de hallar la posición mejor dentro de las que se encuentren similares, pero no parece en particular ser un elemento necesario para así generar una métrica de similitud.

Los autores sin embargo, describen el algoritmo usado, el cual se describe en la siguiente figura 3.3:

Los autores hacen un análisis de sus búsquedas a partir de lo que se ha definido en las páginas anteriores y llega a la conclusión que la dirección de su

[2]Las piezas blancas se representan en mayúsculas y las negras en minúsculas. En el artículo original se utilizan para Rey, Dama, Alfil, Caballo, Torre y Peón, los siguientes símbolos (en inglés): K,Q,B,N,R,P y k,q,b,n,r,p, respectivamente para blancas y negras.

Algorithm 1 Chess Games Indexing and Retrieval

```
1: procedure IndexGames(G)
2:   for each game g ∈ G do
3:     Skip the first numskip moves.
4:     for each move m ∈ moves(G) do
5:       brdenc ← ∅
6:       board ← board matrix on applying move m
7:       for each piece (p, x, y) ∈ board do
8:         brdenc ← brdenc ∪ (p, x, y)
9:         for each (p, x', y', w) ∈ r-closure(p, x, y) do
10:          brdenc ← brdenc ∪ (p, x', y', w)            ▷ Eq. 2
11:        end for
12:        for each α ∈ {a, d, x} do
13:          for each (p, p', x', y') ∈ α-closure(p, x, y) do
14:            brdenc ← brdenc ∪ (p, p', x', y')   ▷ Eq. 3, 4, 5
15:          end for
16:        end for
17:      end for
18:      Add document brdenc to index
19:    end for
20:  end for
21: end procedure
22: procedure Retrieve(Q)
23:   brdenc ← ∅
24:   board ← board matrix of Q
25:   for each piece (p, x, y) ∈ board do
26:     brdenc ← brdenc ∪ (p, x, y)
27:     for each α ∈ {a, d, x} do
28:       for each (p, p', x', y') ∈ α-closure(p, x, y) do
29:         brdenc ← brdenc ∪ (p, p', x', y')
30:       end for
31:     end for
32:   end for
33:   Retrieve from index by executing query brdenc
34: end procedure
```

Figura 3.3: Algoritmo para hallar la similitud de una colección de posiciones dada una referencia a buscar [44].

trabajo puede orientar a otros trabajos futuros considerando lo que nosotros ya sabemos, que las posiciones de ajedrez pueden representarse como gráficas (la configuración de las piezas como nodos y la relación entre ellas como vértices). Esto –dicen los autores– podría incluso a generalizar las búsquedas aproximadas en objetos gráficos.

3.3. Experiencia a través de *Chunking*

En la búsqueda de configuraciones (patrones) ganadoras o de ventaja suficiente, se tiene –particularmente en el ajedrez– un cuerpo de experiencias

pasadas que están ya documentadas en las bases de datos (partidas de ajedrez), que prácticamente contienen toda la historia del ajedrez competitivo. Para la mayoría de los juegos, la fuerza de un jugador, es decir, su habilidad para jugar mejor puede incrementarse revisando encuentros anteriores, particularmente de los mejores exponentes de dicho juego. Esto es aplicable al Go, al Ajedrez y a las damas inglesas, por mencionar algunos juegos de suma-cero.

Este artículo, de Michael George y Jonathan Schaeffer, [45] contiene algunas ideas interesantes. Parte de la idea de que un efecto significativo en los seres humanos para mejorar su habilidad en un juego depende en su capacidad de recordar encuentros pasados, propios o de otros jugadores, o bien, fragmentos significativos de partidas anteriores. Durante una partida un jugador fuerte frecuentemente hace correlación entre la posición (configuración), que tiene en el tablero y en otras que son parecidas, similares o incluso idénticas, que ha visto anteriormente.

En fragmentos de posiciones similares, el jugador puede recordar las jugadas ganadoras, aunque la configuración de las piezas en el tablero no sea estrictamente la misma que conoce anteriormente. Esta capacidad no ha sido implementada en los programas de ajedrez, lo cual podría darles algunas ventajas en la búsqueda de las jugadas ganadoras. Un programa podría ser capaz de extraer estas configuraciones así como los movimientos hechos y así, hacerse de una experiencia similar a la que tienen los maestros de ajedrez.

Hoy en día, gracias a que existen manejadores de bases de partidas (*Chessbase*, *Chess Assistant*), millones de partidas, literalmente, pueden ser consultadas y además, ya están en un formato que es un estándar (llamado *PGN - Portable Game Notation*). Desafortunadamente, no hay una manera de saber *a priori* si una jugada es buena o no. Los investigadores del artículo suponen (y suponen bien), que debe haber alguna manera de extraer información útil de estos archivos enormes de partidas.

Ya Chase y Simon [21] trabajaron sobre la percepción del ajedrecista, los cuales se basaron en los trabajos de Adriaan de Groot [29]. El hallazgo de este último investigador fue encontrar que los ajedrecistas utilizan el concepto de similitud de posiciones basándose en "chunking", que es básicamente separar los elementos de la posición en patrones elementales. Si se pudiese hacer esto en términos de un programa de computadora, se podría sacar ventaja a la experiencia pasada y así aplicar un enfoque más basado en el conocimiento que en un proceso de búsqueda masiva y de procesamiento de jugadas y respuestas.

George y Schaeffer plantean entonces el diseño de MACH (a *Master Advisor for Chess*), en donde el sistema usa una base de chunks definidos para acceder a la colección de partidas de grandes maestros. Cabe decir que los autores indican que estos chunks, que son los patrones o configuraciones más básicas, fueron descritas por maestros de ajedrez. Estos chunks definen patrones de ajedrez que frecuentemente ocurren en las partidas. De esta manera, todas las posiciones de las partidas de los grandes maestros pueden ser analizadas a partir de sus bloques (chunks), constituyentes y organizados de manera tal que puedan ser puestos en una base de datos para un acceso rápido y eficiente. De esta manera MACH puede -cada vez que tiene una nueva partida por analizar- hacer los correspondientes chunks y referenciarlos a la base de datos para así hallar posiciones similares.

Dos temas fundamentales de este artículo son:

- Cómo definir estos bloques elementales (chunks), para así definir las posiciones y configuraciones de interés.
- Cómo podemos reconocer que dos posiciones son similares.

Es claro que los ajedrecistas tienen esta habilidad de determinar las características más importantes de una posición, basándose en sus conocimientos previos, reconociendo las similitudes halladas en otras partidas que han jugado o han visto. Esta capacidad de extraer esta información, como ya demostraron investigadores anteriores [21, 29], se hace en fragmentos, en bloques, dentro del tablero, lo cual es la clave para reconocer otras posiciones similares. Esto ayuda eventualmente al maestro a guiarse sobre cómo debe entonces jugar. Se sabe, por los experimentos de De Groot, que la teoría de los chunks es una de las más aceptables para modelar cómo piensa un maestro de ajedrez. Posnyanskaya y Tikhomirov [67] mostraron que un jugador experimentado analiza primero la posición en el tablero para buscar información que le sea relevante, antes de aplicar una estrategia en particular, todo esto analizando cómo se movían los ojos a través del tablero en el proceso de análisis de las posiciones.

Otro importante trabajo es el realizado por Bratko, Tancig y Tancig [17], quienes hallaron un método para detectar patrones posicionales en ajedrez. Más interesante es saber que Bratko *et al*, hallaron que en promedio, hay unos 7.54 chunks por posición (aunque falta aclarar cómo definen cada chunk). En

un trabajo similar, Hans Berliner (excampeón mundial por correspondencia), y Campbell [20], pudieron derivar chunks de posiciones para poder jugar finales de peones de forma automatizada.

El primer trabajo relevante en este sentido es el de Zobrist y Carlson [84] pues fueron los primeros en modelar la teoría de chunking en el juego del ajedrez. Zobrist y Carlson primero buscaban todas las instancias de ciertos patrones (que serían los chunks), y al final de cuentas todo lo hallado lo guardan en algo que llamaron *snapshot*. A partir de ahí hacían búsquedas para evaluar las posiciones de ajedrez.

George y Schaeffer decidieron entonces crear un lenguaje de descripción de las posiciones, en donde se plantean los elementos fundamentales del tablero. Básicamente deciden las propiedades y características de una posición a partir de los ataques y defensas de las piezas y de la proximidad entre ellas. Definen además algo que llama la atención, el concepto del conjunto de chunks que deben existir y que no son opcionales. También tienen el concepto de las características opcionales, que muchas veces incluso están presentes pero que no son necesarias para que la configuración sea definida. Importante es que se pueden crear chunks basados en los ya reconocidos, los cuales bien podrían llamarse meta-chunks, pero esta notación no la usan los autores. Los chunks deben poder tener parámetros para así evitar en la medida de lo posible las repeticiones sin sentido. Finalmente el lenguaje da un consejo sobre cómo actuar en las posiciones halladas.

De acuerdo con el lenguaje creado por los autores, esto sería un chunk típico (ver figura 3.4):

Los autores entonces analizan una base de partidas y definen los chunks pertinentes de cada una de las posiciones de las mismas. Ellos consideran que dos posiciones son similares si contienen chunks idénticos pero incluso los propios autores se dan cuenta que deberían buscar un criterio de similitud más robusto. Todos los chunks se pre-calculan para posteriormente ponerlos en una base de datos y acceder a ellos fácilmente.

La idea del lenguaje que usan los autores es natural, pero no parece haber una justificación para definir los requerimientos obligatorios y los opcionales. Al ser opcionales, por ejemplo, quiere decir que los chunks no se alteran en términos generales, por lo que da la impresión que sobran.

La métrica de similitud de los autores suena razonable pero depende si el lenguaje de descripción es lo suficientemente rico para poder entender las similitudes entre posiciones a partir de criterios simples como el que usan los autores (dos posiciones son similares si tienen los mismo chunks de forma

```
chunk qgdcenter wrt white.                          #El chunk qgdcenter (centro del gambito de dama
                                                    #declinado)  desde el lado de las blancas
classification queens_pawn_opening.                 #clasificación: aperturas de peón dama
require   (WP on D4) & (BP on D5).                  #se requiere que exista un PB en d4 y un PN en d5
require  (WP on C4) & (BP on E6/C6).                #se requiere que exista un PB en c4 y un PN en
                                                    #c6/e6
require  (WP on E2/E3).                             #se requiere un PB en e2/e3
WN on C3.                                           #CB en c3
BN on F6.                                           #CN en f6
WN on F3.                                           #CB en f3
Advice   PutRooks(white,C1,D1), Outpost(white,E5), Outpost(black,E4).

# el consejo es poner las torres blancas, ya sean en c1 o d1; colocar una pieza blanca en la casilla fuerte
e5 y ver otra casilla fuerte
#ahora para las negras, en e4.
end qgdcenter.
```

Figura 3.4: La descripción de un chunk

idéntica).

Parece claro que estamos ante el problema de representación del conocimiento ajedrecístico para la definición de los chunks básicos. El artículo en realidad no explora más que una faceta de estos chunks en posiciones de apertura, pero no considera posiciones o configuraciones del medio juego o finales, y esto pareciese ser la prueba para saber si el lenguaje definido es lo suficientemente expresivo para lidiar con cualquier etapa de la partida. En cualquier caso, algunas ideas sobre el tema de patrones ganadores son importantes y hay que tomarlas en cuenta.

3.4. CHE: Un lenguaje gráfico para expresar conocimiento en ajedrez

Chris Donninger [33] es el autor de Nimzo, un programa de ajedrez que en los años noventa del siglo pasado muestra una serie de avances importantes en el desarrollo de motores de ajedrez. En 1995-96 el autor decide reformular el problema, pues su sistema, aunque jugaba muy bien, le faltaba el "punch" de otros programas como Fritz. Entre las consideraciones importantes es el poder alimentar conocimiento ajedrecístico de alguna manera fácil. Donninger tenía con él un experto en ajedrez, pero no en cómputo y entonces, el autor desarrolla un lenguaje basado en algunos conceptos de Forth que a

decir de él mismo, puede aprenderse en un par de horas, y que permite alimentar información al motor de Nimzo para así darle más fuerza y potencia a sus movimientos.

Donninger indica que un lenguaje que pueda especificar conocimiento ajedrecístico requiere de:

- Una interfaz gráfica en donde el experto pueda alimentar la información de manera sencilla, usando menúes, botones, listas. No debe tener necesidad de programar el sistema.
- Debe poderse, de forma alterna, programar al estilo tradicional, como en cualquier lenguaje de programación.
- El conocimiento adicional no debe modificar el código del programa de ajedrez
- El sistema no debe hacer más lento (idealmente) la ejecución del programa de ajedrez
- El sistema debe poderse extender fácilmente.

La interfaz gráfica diseñada es muy parecida al programa que permite introducir una posición para ser analizada por Nimzo. Donninger habla de dificultades para expresar las relaciones lógicas "not" y "or" y el resolver sus posibles ambigüedades. Una vez que el experto define la posición (en este caso un patrón específico de piezas y/o peones), se le pide le dé un nombre a dicha posición y del cómo se resuelve, es decir, cómo el jugador en cuestión saca ventaja. Esto se convierte al lenguaje de programación CHE, el cual puede después editarse si se desea con "vi" o cualquier otro editor. De hecho, la interfaz gráfica permite que el experto en ajedrez se libere de saber de código en un lenguaje de alto nivel y en ese sentido califica la interfaz como una especie de pre-procesador.

El lenguaje CHE definido por Donninger está influenciado por C y por Forth en su nivel conceptual y filosófico (pág. 236). El lenguaje tiene tres unidades semánticas y sintácticas: "pattern" (patrón), "advice" (consejo) y "rule" (regla).

"Pattern" define las características de la posición (la formación de los peones, la posición del rey, si las damas están en el tablero, etcétera). Un **"advice"** se usa para expresar un plan (cambiar los alfiles blancos, atacar la

cadena de peones con c5, empezar un ataque de las minorías). Los "patterns" y "advices" se combinan en una **"rule"**. Ésta checa primero si todos los patrones de la regla son ciertos, es decir, si se cumplen todas las condiciones impuestas. Si es así, entonces se ejecuta el consejo (advice). Nimzo entonces juega de acuerdo a los consejos que da la regla.

Donninger dice que un patrón y un consejo pueden ser usado dentro de un número arbitrario de reglas. Es posible anidar patrones y consejos, pero no reglas. Así pues, el patrón C podría definirse como el patrón A y el patrón B (o también como el patrón A o el patrón B).

CHE soporta, como en el lenguaje C, una instrucción "include" de manera que patrones y consejos específicos para ciertas aperturas como la Española, India de Rey, Siciliana, pueden ser usados selectivamente.

Cabe decir que anteriormente se han intentado diseñar lenguajes de propósito específico para el ajedrez, como el de George y Schaeffer [45] (descrito en este capítulo). Normalmente este tipo de lenguajes están muy ligados a la aplicación y por ende solamente los programadores pueden extender sus sistemas. En el caso de CHE cualquiera puede cambiar lo que se quiere que haga el motor de ajedrez. De esta manera, dice Donninger, es más fácil así desarrollar y mantener el código existente.

CHE es un intérprete, en la mejor tradición de Forth -de acuerdo al autor. Se produce código p de un sólo byte para una máquina de stack (como en Forth). El intérprete puede analizar unas 1000 líneas de CHE por segundo. Toda esta información pasa al corazón del motor de ajedrez, llamado *Oracle*, a partir del trabajo de Hi-Tech de Hans Berliner [11].

De acuerdo con el autor, el intérprete de CHE en su programa Nimzo es lo suficientemente rápido y no incide en los tiempos de análisis de las variantes que tiene que ejecutar el programa en su poda alfa-beta. El código interpretado es tan rápido como el código de máquina del compilador Watcom C. La razón es que el código p, de un solo byte por instrucción, es muy compacto.

Donninger promete en su artículo que planea implementar la parte de patrones de CHE como un lenguaje de búsqueda de posiciones. Esto podría incrementar el potencial del desarrollo del sistema. Después de definir el patrón, el usuario podría buscar en una base datos para hallarlo. Pero nada de esto se hizo finalmente. La definición de un patrón, el consejo y la regla puede verse así. En la figura 3.5:

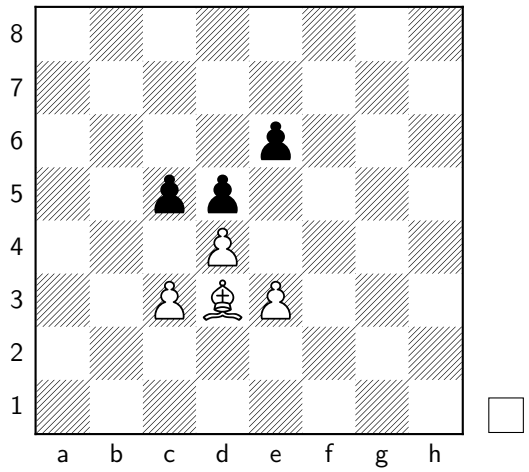

Figura 3.5: **Una configuración (chunk) específica del gambito de dama**

Podría especificarse en CHE de la siguiente manera:

```
pattern pColle
{ wpawns (e3 d4 c3) bpawns (e6 d5 c5) piecesqrs (wB d3) } {};
advice aColle
goodmvs(bP c5d4) goodmvs(bN b8c6) weakmvs(bP c5c4)
rule Colle { pColle} a Colle;
```

Figura 3.6: La configuración Colle

La regla se lee así: si los patrones entre las llaves son verdaderos, entonces, ejecútese el consejo. Así, Nimzo juega primero Cc6 y luego cxd4.

Los resultados, de acuerdo a Donninger, no son demasiado halagadores, sobre todo cuando los programas juegan contra humanos. Dice el autor que en los torneos entre computadoras, hay un balance mutuo de estupidez, pero con los seres humanos no, porque los ajedrecistas poseen un bagaje de conocimientos posicionales que hacen ver de maneras más sutiles algunas posiciones. De acuerdo al autor, y sorprendentemente, “Mientras menos sabe

un programa que juega al ajedrez, mejor es contra los seres humanos" (esta es la filosofía del programa *Fritz*).

En conclusión, la idea de un lenguaje que pueda expresar conocimiento ajedrecístico es interesante pero quizás no es aplicable directamente para cambiar el comportamiento de un programa de ajedrez. De hecho, Donninger piensa que no es "la bala mágica" que se esperaba, al menos en principio. Sin embargo, puede verse como una herramienta que puede incrementar en un número razonable las jugadas que la computadora puede hacer.

3.5. Algunas ideas para un compilador de ajedrez

La representación de conocimiento en los juegos de suma-cero parece ser una parte fundamental para poder entender las configuraciones y/o patrones ganadores. Por ello mismo, el artículo de Clark [23] da una idea interesante, que no se ha llevado a cabo en la práctica, en donde plantea un lenguaje específico para el desarrollo de programas que jueguen al ajedrez en particular.

Clark inicialmente plantea dos de los programas de ajedrez que marcaron la pauta de muchos otros, el de Greenblat (1967) y el de Atkin y Slate (1969), los cuales se basan estrictamente en el artículo desarrollado por Claude Shannon en 1950 [75]. El análisis se basa en una función de evaluación y un algoritmo Minimax que halle la mejor jugada de acuerdo a determinada profundidad. La función de evaluación de un programa de ajedrez de ese entonces era muy limitada, sin embargo, los programas mencionados podían estar alrededor del lugar 500 de los jugadores de ajedrez registrados en el Reino Unido. No es esto, desde luego, algo sorprendente. De hecho, ambos programas califican como un aficionado con muy poca experiencia.

El autor menciona el programa de Samuel, que competía en las damas inglesas contra el campeón del mundo. Y aunque el trabajo de Samuel es importante, el problema de jugar a las damas inglesas se resolvió apenas hace unos años [74]. En cualquier caso es una referencia obligada pues muchas de las técnicas usadas en los programas de ajedrez tienen un paralelo con lo que Samuel intenta en su momento.

Un punto importante y quizás un parteaguas en el estudio de la representación de las posiciones de ajedrez (ganadoras o no), parte del trabajo de de Groot [29], en donde le pide a una serie de jugadores, entre maestros y

aficionados, que recordaran posiciones a las que se les mostraba por pocos segundos. Desafortunadamente, dice Clark, los resultados de De Groot son cuantitativos y no se analiza cómo un jugador fuerte sabe de las relaciones entre las piezas, lo que bien podría dar algunas pistas a una representación del conocimiento en las posiciones en ajedrez.

Por otra parte, Mijaíl Botvinnik (excampeón mundial y primer campeón mundial de la posguerra (1948)), trabaja en un programa de ajedrez, en la Unión Soviética, en donde construye una descripción, la cual llama un mapa matemático de la posición, basándose en las casillas que las piezas atacan en la una o más jugadas [14]. Sin embargo, no continúa esa idea y parece abandonarla por un enfoque más convencional.

Todo lo anterior sugiere que no resulta una mala idea diseñar un lenguaje de programación orientado al ajedrez, el cual Clark basa en Algol 60, al cual le llamó Algol 64. Cabe enfatizar que dicho lenguaje no se ha escrito y que sólo es una propuesta, pero parece ser un camino a explorar, considerando que hay que representar de alguna manera las posiciones en el tablero de ajedrez.

De acuerdo al autor, un programa escrito en Algol 64 debiese ser un algoritmo para encontrar una jugada en una posición dada de ajedrez. Se esperaría que un jugador más fuerte pudiese escribir un programa que jugase mejor que el que pudiese escribir un jugador débil. Cabe señalar que la idea de Clark es utilizar el modelo de Algol para escribir un pre-procesador que tome como programa fuente en Algol 64 y lo convierta a Algol 60. Por ello mismo, el usuario podría, en Algol 64, poder dar nombres a las variables, tener tipos real, booleano, entero, flotante, y poder manipular finalmente esto en expresiones, en funciones y procedimientos.

Clark usa la notación algebraica para el ajedrez, la cual es la oficial de la Federación Internacional de Ajedrez (FIDE, por sus siglas en inglés), y abrevia las piezas como BN para indicar que hablamos de un caballo negro (*Black Knight*) y WB, para representar a un alfil blanco (*White Bishop*). El autor define entonces [23]:

A partir de esto se generan listas de variables del mismo tipo, referidas como *squarelist*, *movelist*, etcétera. Clark define un vector como una lista de tipos mixtos que describen una posición. Sin embargo usa la sintaxis de Algol 60 aunque cambia el símbolo de igualdad por "is", (como en algunos Prolog), para mejorar la legibilidad del código.

El lenguaje tiene instrucciones como **for** $<$ *variablename* $>$ **in** $<$ *listname* $>$ **do**

Tipo	Rango de valores
rank (fila)	1-8
file (columna)	a-h
square (casilla)	a1 ... h8
move (movimiento)	la combinación de dos casillas
position (posición)	todas las posibles configuraciones: WP, BK, WQ, etcétera
piece	una pieza, que puede ser blanca (White) o negra (Black) o incluso una casilla desocupada
colour	White (blanco), Black (negro)
integer	-n, ... , -3, -2, -1, 0, 1, 2, 3 ... n
predicate (predicado)	true, false

Figura 3.7: Las primitivas del lenguaje de propósito específico para el ajedrez

No obstante, no declara operadores aritméticos y si acaso se requiere hacer alguna operación aritmética, se hace llamando a procedimientos. El autor además pone detalles sintácticos para las variables y las constantes. Por ejemplo, los nombres definidos por el usuario deben aparecer todos en mayúsculas mientras que las constantes y las palabras reservadas se ponen en minúsculas, usando como separador el guión bajo. Esto hoy en día no se necesita hacerlo.

Clark define también los siguientes procedimientos y funciones:

squareprocedure	FROM(MOVE)	La casilla desde donde se mueve una pieza
movelistprocedure	MOVESTO(SQUARE, POSITION)	Lista de los posibles movimientos desde una casilla
colourprocedure	COLOUR(PIECE)	auto-explicativa
pieceprocedure	PIECE(SQUARE,POSITION)	la pieza en una casilla (SQUARE)

Figura 3.8: Los procedimientos del lenguaje de propósito específico para el ajedrez

El autor propone la creación entonces de un pre-procesador que pueda traducir el código de Algol 64 al de Algol 60 para finalmente poderse compilar. Sin embargo, Clark propone algo que es de suma importancia y que no se ha hecho (o al menos en forma definitiva, aunque hay esfuerzos como [45, 33]): la

generación de un lenguaje especializado para el ajedrez podría ser usado para comunicar algoritmos de ajedrez, evitando así la duplicación de esfuerzos.

Un análisis detallado del artículo de Clark muestra que deja algunos puntos sin resolver, asumiendo que su implementación en el lenguaje que propone es trivial, pero en la práctica puede hallarse que esto no es así de simple.

Por otra parte, la evolución de los lenguajes de programación hacen francamente innecesaria la idea de un pre–procesador que traduzca de Algol-64 a Algol-60 para así compilar el código. No obstante la idea de Clark de definir un conjunto de instrucciones para escribir programas de ajedrez es muy interesante y evidentemente puede ampliarse.

3.6. "El Sistema" (Ajedrez Dinámico)

Hans Berliner se hizo campeón mundial por correspondencia mucho antes de que las computadoras atacaran el problema del ajedrez. Es además el creador del programa HiTech, que en los años setenta del siglo pasado participa con éxito en diversas competencias, incluso humanas. Berliner ha escrito un libro muy original [11], en donde en su tercer capítulo habla de ajedrez dinámico e introduce la noción de *chunks* pero enriqueciéndole con la idea del dinamismo de las posiciones en donde se dan.

El autor habla de los experimentos de principios de los años 1930, los cuales son replicados más adelante por Adriaan de Groot [29]. En ellos se demuestra básicamente que los jugadores de ajedrez mantienen una colección de configuraciones, que les guían para hallar las mejores jugadas. La diferencia entre un jugador fuerte y uno débil se basa en la cantidad de patrones, chunks conocidos, los cuales son aplicables en una diversidad de posiciones.

De acuerdo a Berliner, la naturaleza del cerebro humano es tal que solamente puede tener hasta siete chunks sin perder ninguno (por sobrecarga), aunque no aclara de dónde obtiene semejante cifra. Pero asumiendo que Berliner tenga razón, hay entre 2 y 7 chunks presentes en la interpretación del tablero en cierto nivel. Esto –dice el autor– es esencialmente asignarle una *firma* a la situación percibida. De esta manera se puede clasificar y ver sus propiedades de las situaciones que tienen la misma firma. En ajedrez, por ejemplo, hay –ya sabemos– ciertos chunks muy conocidos: la seguridad del rey, la conformación de los peones en el tablero, el alcance de las piezas, el dominio del centro del tablero, etcétera. Hay cientos de miles de chunks, los cuales permiten al jugador de ajedrez entender cómo orientarse en la

diferentes posiciones que se le presentan.

3.6.1. Tipos de chunks

Berliner muestra algunos chunks básicos:

- Cuáles son las mejores piezas menores.
- Cómo cooperan las piezas.
- Puntos fijos para el ataque.
- Qué está bien defendido y qué no.
- Complejos de casillas de determinado color.

El autor muestra cómo reconocer chunks, y los describe como **Chunk (Faceta o Casilla; Lista de las casillas de las piezas involucradas)**. Al igual que otros autores, Berliner indica que la parte crítica que hace que las piezas sean parte de un chunk son las relaciones de ataque y defensa entre las piezas que hay en el chunk. Por ejemplo, véase la figura 3.9:

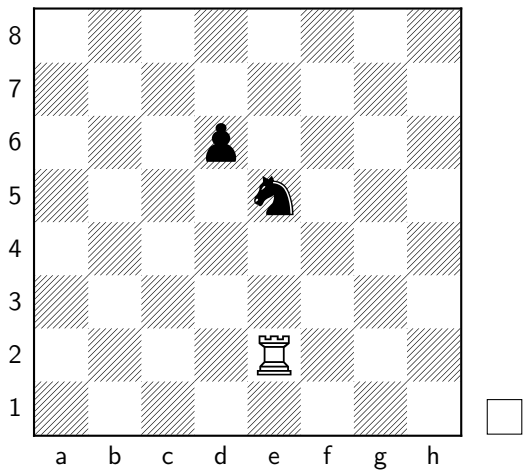

Figura 3.9: **Un ejemplo de un chunk que involucra tres piezas**

Este es el chunk que presenta Berliner: Chunk(e5; W: e2; B: e5, d6), en donde el caballo negro es el objeto, la torre banca en e2 lo ataca, pero

el peón de d6 lo defiende. Estas tres piezas conforman el chunk básico. Es interesante sin embargo hacer notar que Berliner no hace mención de las piezas involucradas en el chunk y da la impresión que esta información es relevante y que debería estar, pero Berliner por alguna razón no la pone.

Hay muchos tipos de chunks, de táctica (en donde una combinación de movimientos obligados decide la ventaja, en general), o posicionales, donde se observa la fuerza de las piezas por las configuraciones de peones. Estas configuraciones pueden ser muy complejas, como por ejemplo, el Chunk "fortaleza", en donde las blancas no pueden hacer progreso alguno para ganar, a pesar de su ventaja material. (ver figura 3.10):

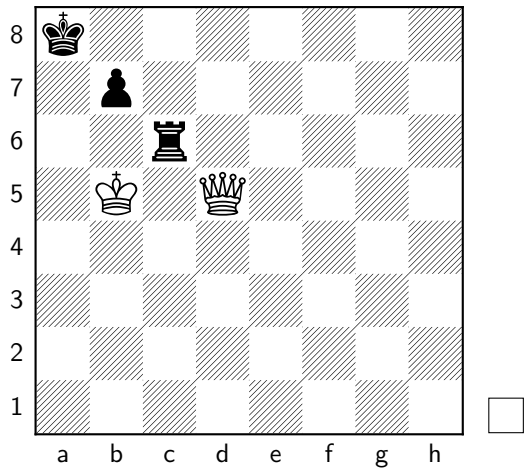

Figura 3.10: **El chunk fortaleza**

Berliner indica que los chunks por sí mismos son útiles, pero que en la medida que se entiende la dinámica de los mismos, resultan mucho más aplicables. Esta dinámica plantea el concepto de interacción de chunks. De acuerdo al autor **El entendimiento correcto de la interacción de los chunks nos lleva directamente a las estrategias**. Berliner define el espacio de abstracción, donde se trabaja sobre las diferentes posibilidades futuras. En el caso del ajedrez, no es el espacio de posibles jugadas, sino el espacio al que pueden ocupar piezas y peones en el futuro. Esto es –dice Berliner– el espacio de la planeación estratégica.

Las conclusiones del autor son:

1. Los chunks son entidades que apuntan sobre la manera de proceder estratégicamente.

2. Los chunks pueden interactuar con otros chunks para cambiar la evaluación que se aplicaría si solamente existiese uno de ellos presente.

3. Hay muchas estrategias que gobiernan los ataques (la seguridad del rey, tener ventaja en el desarrollo, una estructura de peones débil, etcétera).

3.6.2. Un ejemplo completo de la evaluación de un chunk

Berliner plantea la siguiente posición en la figura 3.11:

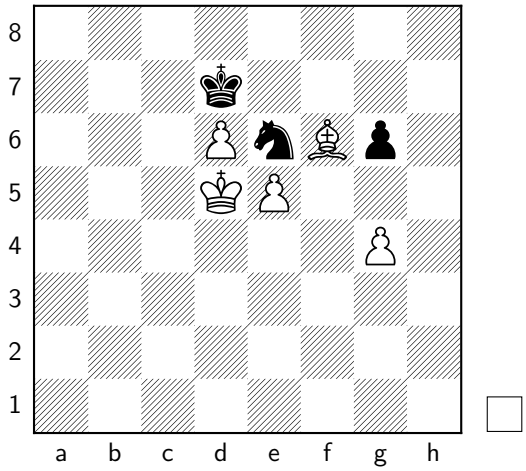

Figura 3.11: **¿Pueden ganar las blancas?**

Aparentemente la respuesta es afirmativa. Tienen dos peones de ventaja. Sin embargo, tienen un alfil de casillas negras que aparentemente no puede ayudar al triunfo. El chunk se define así: chunk(peones centrales; W: d6, e5, f6, d5; B: d7, e6), chunk(peones en la columna g; W: g4, d5, f6; B: g6, e6). Si las blancas quieren ganar, pueden intentar dos posibles planes:

1. Tratar de avanzar los peones centrales, en particular el peón de e5. Sin embargo esto falla porque el caballo negro puede ir a 8 diferentes casillas

las cuales no pueden ser atacadas por el alfil, el rey y los peones. Así, mientras el rey negro se mantenga en d7 y el caballo se mueva hacia adelante o hacia atrás, no se pueden avanzar los peones blancos.

2. Tratar de infiltrarse con el rey por g5 para ganar el peón de g6. Esto involucra llevar el rey hasta h4 desde donde podría infiltrarse. El caballo es el único guardían de g5 por lo que debe quedarse en donde está. Pero al mover el rey blanco, entonces el rey negro está libre de moverse hacia adelante y hacia atrás.

Berliner llega a la conclusión que esta posición es un empate y considera que ninguna computadora en el mundo puede llevar a la conclusión de que esta posición no se puede ganar por parte de las blancas[3]. Este ejemplo muestra el poder de la idea de los chunks.

3.7. Un programa de ajedrez que hace *chunks*

Murray Campbell y Hans Berliner [81] proponen un programa que usa conocimiento en forma de chunks para lograr el éxito, es decir, jugar bien una parte de la partida, o una posición determinada. Para ello, tomaron un subconjunto de los finales de reyes y peones que ha sido estudiado por más de 300 años. CHUNKER es el software que tienen una biblioteca de instancias de chunks en donde cada uno de ellos tiene una lista de propiedades y cada instancia tiene un conjunto de valores para estas propiedades. Esto significa que CHUNKER puede "razonar"[4] sobre la posición y hacer una búsqueda sobre la misma que, de otra manera, ésta [la búsqueda], tendría que hacerse con más recursos o buscar más de una vez.

El programa resuelve el problema más difícil en el dominio mencionado, en donde se requieren 45 ply[5] de búsqueda[6]. CHUNKER lo resuelve en 18 ply y

[3]Cabe señalar que el programa Komodo 9.02, de los más fuertes del mundo al momento de escribir esto, indican ventaja ganadora pero todas las variantes que muestra no tienen progreso alguno para la causa blanca.

[4]Del artículo original.

[5]El término *ply* se usa en el ajedrez por computadora como un movimiento de uno de los jugadores. Una jugada completa consiste en dos movimientos, el de blancas y negras. (https://chess24.com/en/read/glossary/ply)

[6]Calculado en su momento como 10^{13} años de tiempo de CPU en los sistemas de ese año (1983), indican los autores.

un minuto de CPU. También señalan los autores del artículo que CHUNKER ha descubierto dos errores en la literatura del problema y que h encontrado un nuevo teorema (en este dominio), que permite evaluar las posiciones con un nuevo grado de simplicidad y confianza.

El dominio del estudio es los finales de reyes y peones, pero particularmente tres peones contra tres peones, como se muestra en la figura 3.12:

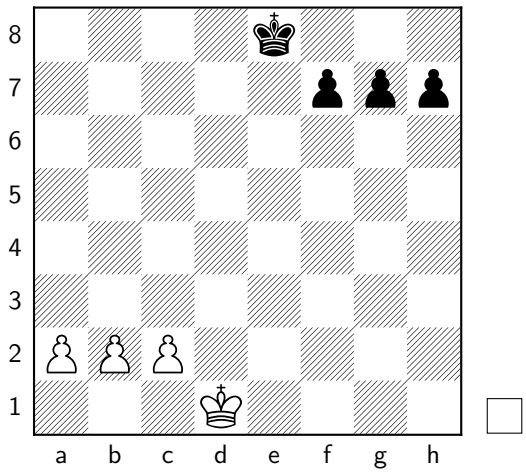

Figura 3.12: **Una posición en el dominio**

- Los autores indican que categorizar en chunks esta posición se puede hacer de manera directa.
- Es un final de partida no trivial e incluso se sabe de las dificultades de jugadores fuertes que no conocen los detalles de la complejidad de este final.
- Un programa que use una tabla para hallar las jugadas correctas requeriría 2^{29} entradas[7].

Murray y Berliner (el primero uno de los artífices de la máquina Deep Blue de IBM y el segundo Maestro Internacional) indican que para resolver este final se requiere aislar los diversos chunks que hay en la posición, indicando

[7]Las tablas de Lomonosov contemplan solamente los finales de hasta siete piezas.

que esto hace además que la biblioteca de chunks sea manejable. Por ejemplo, toman el siguiente chunk (figura 3.13):

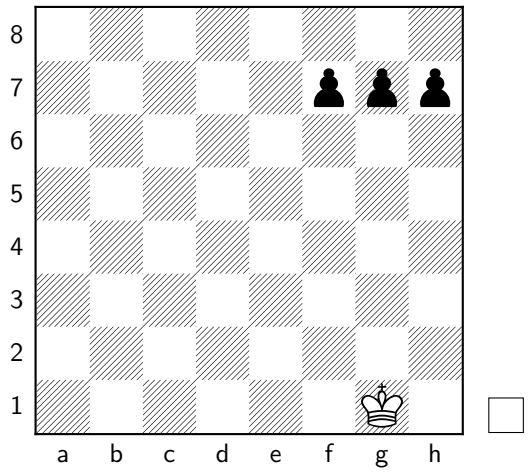

Figura 3.13: **Un chunk de la posición a analizar**

El diagrama anterior presenta un primer chunk, pero hay varios que pueden verse a lo largo del desarrollo de la partida, del final que se está analizando. Considerando esto, es decir, todos los posibles chunks, el método de análisis de CHUNKER es una búsqueda sobre el árbol de variantes a lo ancho limitado en la profundidad, con poda alpha-beta en donde los nodos terminales son aquellos que ocurren cuando un peón se convierte en dama, en jaque o posiciones de ahogado o tablas por repetición de movimientos. Las posiciones no decidibles en la profundidad máxima se consideran de empate o igualdad.

CHUNKER resuelve satisfactoriamente el final, pero a pesar de las conclusiones de los autores, en donde se dice que tienen un método para que un programa manipule las propiedades de los chunks en una biblioteca para evaluar todas las posiciones de ajedrez. Sin embargo, es claro que todas sus pruebas se hacen sobre un subdominio del ajedrez, el cual en términos reales no es demasiado complicado a pesar de las sutilezas del final presentado.

El problema que el sistema presenta es la falta de generalización para otro tipo de posiciones con más piezas, en donde el problema se incrementa si se trata de analizarlo de acuerdo a las ideas presentadas. No obstante, es claro que la parte de chunks tiene una serie de particularidades de las cuales

bien podrían sacarse ventaja para hacer más visibles las posibilidades de un programa que hace chunks para jugar al ajedrez.

3.8. *Priyomes* o patrones en la estructura de peones

Un "priyome" es un concepto ruso para las posiciones estratégicas que ocurren frecuentemente. Es el equivalente a los patrones que se dan muchas veces en posiciones que se resuelven tácticamente. En un priyome no solamente tenemos una estructura estática, es decir, la posición de los peones y las piezas, sino que además se asocian maniobras que explotan una estructura en particular.

El gran maestro Andrew Soltis [76] introduce la idea de los priyones. Indica que en muchos casos hay priyomes tan populares que ni siquiera están definidos específicamente. El entrenador ruso Anatoly Terekhin estima que un maestro debe conocer unos 100 priyomes y desde luego, los grandes maestros conocen muchos más. Esto habla del conocimiento de lo que ocurre en el tablero de ajedrez.

El primer priyome que menciona Soltis es en realidad creación de Henry Bird, jugador del siglo XIX el cual concibe una estructura de peones que busca explotar el control de la casilla e5 para así obtener ventaja. En la mencionada partida se llega a la siguiente posición (figura 3.14):

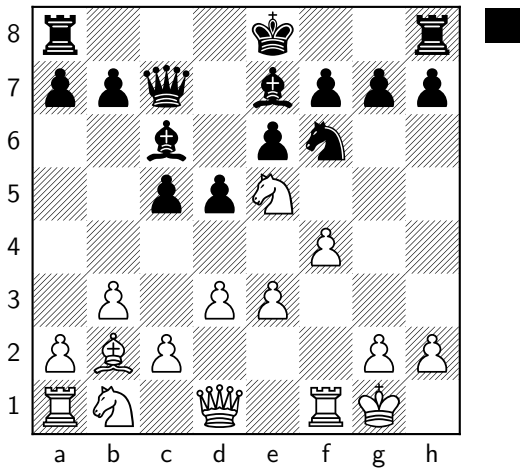

Figura 3.14: **Bird, H.–Riemann F.**, 1885. Juegan las negras

Soltis indica que el plan del blanco es seguir con Cd2, De2 y eventualmente e3-e4 y/o c2-c4, con posición preferible. El priyome es muy importante porque tanto jugadores de la talla de Nimzowitsch o Fischer, han adoptado con los colores cambiados en la defensa Nimzo–India después de **1 d4 ♘f6 2 c4 e6 3 ♘c3 ♗b4** con la idea de **4. . . ♗×c3** y **5. . . ♘e4**.

Soltis repasa otros 24 priyomes que son muy frecuentes, pero la concepción es la misma: se trata de posiciones cuya estructura de peones está muy bien definida y en donde se puede adquirir un mejor juego, la ventaja, vía una maniobra que en ocasiones no tiene visos de ganar material sino el controlar una casilla en particular, para así asegurarse la ventaja en el futuro.

Cabe decir que los priyomes, a pesar de ser conocidos en la literatura especializada en ajedrez, no se han convertido en una herramienta para catalogar la ventaja en muchas posiciones de ajedrez. No existe de hecho ningún formalismo asociado a esta idea, a pesar de la bondad de la misma.

3.9. Otros enfoques

Otros trabajos que vale la pena mencionar: "Templates in chess memory: A mechanism for recalling several boards" [48]. En este artículo se analiza de manera empírica y teórica, lo referente a los *chunks* de memoria [45] y

se sugiere que los ajedrecistas usan estructuras de almacenamiento de largo plazo, a la que llaman "templates" que vendrían a ser el siguiente paso en los chunks.

Gadwal *et al* [32] investiga un enfoque basado en conocimiento para construir un prototipo de un tutor de ajedrez, lo que podría ayudar a los estudiantes a aprender cómo jugar cierto tipo de finales de alfiles. La idea podría ser usada en el lenguaje de representación que se define más adelante, para eventualmente construir patrones de más alto nivel para el aprendizaje de los patrones básicos de táctica.

Por otra parte, P. Ciancarini y M. Gaspari [22] usan un enfoque muy interesante para definir un tipo de interfaz para las posiciones del medio juego a través de una base de conocimientos. Esto lo observamos como una idea complementaria al lenguaje de descripciones que se define en este trabajo. Sin embargo, la motivación de los autores es diferente al enfoque aquí tratado, pero puede ser usado para intentar construir una base de datos basada en conocimiento para patrones de ajedrez específicamente.

3.10. Resumen

En este capítulo se hace una revisión de los artículos más importantes sobre los juegos de suma–cero e información perfecta, en particular para el caso del ajedrez. Se analizan los casos, de forma histórica, los trabajos en ajedrez, empezando con el de Zobrist y Carlson y analizando un trabajo relacionado con las posiciones similares en el juego del ajedrez. Se introduce el concepto de *chunking*, revisando los artículos de Chris Donninger y Hans Berliner. También se revisa el artículo de Clark, que propone la idea de un lenguaje de programación cuyo dominio sea el ajedrez. Finalmente se añade el concepto de "priyomes", que son usados por la literatura de ajedrez rusa para describir patrones conocidos en ajedrez en posiciones específicas para adquirir la ventaja.

Capítulo 4

Patrones para el caso de algunos juegos de suma–cero e información perfecta

Los patrones pueden ser una interesante herramienta para usarse en juegos de suma–cero como el ajedrez, gato o NIM, entre otros. En este capítulo se discuten algunos de ellos a partir de las características que se presentan en diferentes situaciones conocidas dentro de cada juego.

4.1. Introducción al juego del gato

El juego del gato es un juego de lápiz y papel entre dos jugadores: O y X, que marcan los espacios de un tablero de 3×3 alternadamente. El juego del gato es relativamente trivial y su análisis puede dar lugar a patrones que muestren las diferentes posibilidades para los jugadores. Cabe señalar que en las siguientes definiciones de patrones del juego del gato, se puede sacar ventaja de la simetría del juego mismo.

4.2. Los patrones básicos del juego del gato

El juego del gato es muy sencillo. Definimos tres posibilidades para hacer un movimiento dentro del tablero:

- Tirar en una esquina.

- Tirar en el centro del tablero.
- Tirar en una no–esquina[1].

A continuación se presentan algunos patrones para el juego del gato:

4.2.1. Inicio en una esquina

[**También conocido como**] - Mejor primer movimiento.
[**Contexto**] - Inicio del juego (tablero vacío). La mejor jugada del jugador que inicia (las "X"), es en una esquina del tablero del gato.
[**Problema**] - Hallar la mejor jugada inicial en el tablero del gato para el primer jugador.
[**Solución**] - Ejecutar tirar en una de las cuatro esquinas, que es donde se tienen 4/5 de probabilidades de ganar.
[**Estructura**] - Este patrón permite definir los siguientes movimientos del primer jugador, los cuales –considerando el mejor movimiento por cada jugador– se gana en 4 de los 5 posibles movimientos del contrario (las "O"), y de nuevo, esto se aplica por la simetría del juego.
[**Ejemplo resuelto**] - La figura 4.1 muestra la secuencia a realizar.

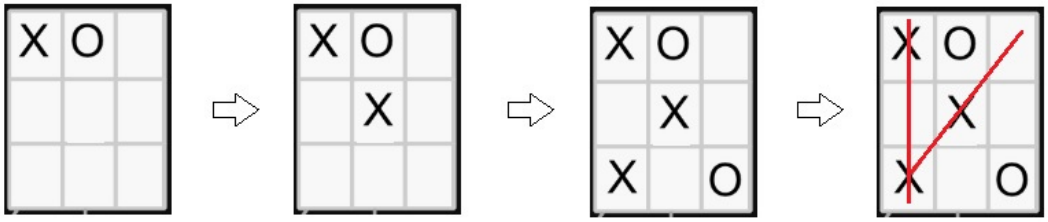

Figura 4.1: El mejor movimiento para el jugador que inicia, las "X".

4.2.2. Responder en el centro

[**También conocido como**] - Mejor respuesta del jugador "O".
[**Contexto**] - La única jugada que empata a cualquier inicio del jugador "X".

[1]Una no–esquina es el conjunto de las casillas medias al lado de la casilla central.

[**Problema**] - Hallar el mejor movimiento en el tablero del gato para el segundo jugador es su primer movimiento.
[**Solución**] - Ejecutar el movimiento en el centro del tablero reduce a cero las posibilidades de perder (considerando que cada jugador hace el mejor movimiento en cada oportunidad).
[**Estructura**] - Al responder en el centro, el juego se reduce a dos tipos, una partida con un primer movimiento en las esquinas o bien, una partida con un primer movimiento en las no-esquinas.
[**Ejemplo resuelto**] - La figura 4.2 muestra la secuencia a realizar.

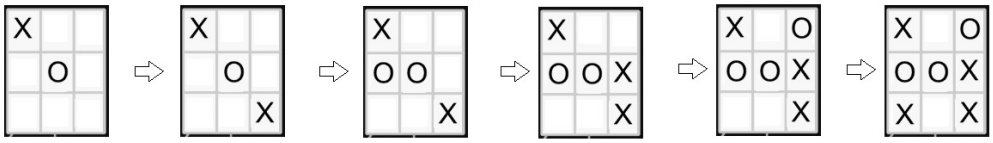

Figura 4.2: El mejor movimiento para el segundo jugador, las "O".

4.2.3. Tirar en una no–esquina

[**También conocido como**] - Confundiendo al jugador "O".
[**Contexto**] - Iniciar con una jugada con menos probabilidades de triunfar para desorientar al rival.
[**Problema**] - Buscar confundir al rival sobre las mejores jugadas de "X".
[**Solución**] - Ejecutar el movimiento en una no–esquina del tablero reduce a 2/5 las posibilidades de ganar (considerando que cada jugador hace el mejor movimiento en cada oportunidad).
[**Estructura**] - Al responder en el centro (por parte de las "O"), el juego básicamente es un empate, considerando que ambos jugadores hacen sus mejores movimientos.
[**Ejemplo resuelto**] - La figura 4.3 muestra la secuencia a realizar.

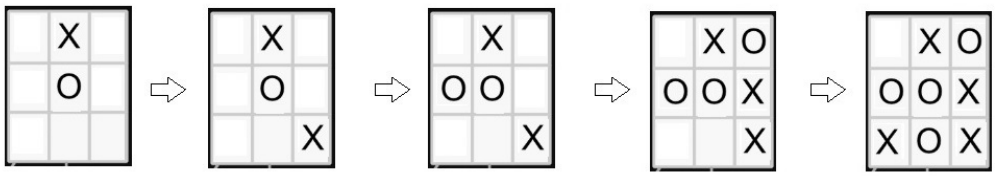

Figura 4.3: Iniciar en una no–esquina tiende al empate.

Estos patrones muestran que en el gato se pueden definir las posiciones ventajosas incluso prácticamente al inicio de la partida. Así entonces, si uno de los jugadores llega a uno de estos patrones, el otro jugador sabe cómo actuar para obtener la ventaja (en este caso, el triunfo en el juego). Dicho de otra forma, no hay necesidad de usar Minimax para saber cuál es la mejor jugada.

4.3. Patrones en el juego de NIM

El juego de NIM está resuelto matemáticamente (ver Capítulo 5.5.2). Sin embargo, cotidianamente los jugadores en NIM no usan la *suma NIM* para hallar la mejor jugada. En lugar de eso hacen uso de patrones que se saben ventajosos y que, al hacer las jugadas correctas, se logra obtener finalmente la ventaja ganadora y por ende, en triunfo.

En NIM, usando el algoritmo hallado por Boulton [16], encontramos 18 patrones ganadores y de hecho, la posición inicial es una posición que da ventaja por lo que el jugador que le toca jugar a partir de esta posición, está en desventaja. Estos son todos los patrones ganadores: 1-3-5-7, 2-5-7, 3-4-7, 1-2-4-7, 3-5-6, 1-2-5-6, 1-3-4-6, 2-4-6, 1-1-5-5, 5-5, 1-4-5, 1-1-4-4, 4-4, 1-1-3-3, 3-3, 1-2-3, 1-1-2-2, 2-2, 1-1-1.

4.4. Algoritmo ganador en NIM usando patrones

Para poder ganar en el juego de NIM, se parte de la posición inicial 1-3-5-7 y de acuerdo al movimiento que haga el rival, buscamos un patrón que sea de nuevo ventajoso, para que de nuevo el adversario vaya llegando a la

situación límite, que básicamente son los patrones 1-1-5-5, 1-1-4-4, 1-1-3-3, 1-1-2-2, 5-5, 4-4, 3-3, 2-2, 1-2-3, 1-1-1.

La solución completa para cualquier jugada que haga el adversario (partiendo de la posición inicial), se analiza en 5.5.4. Sin embargo, se puede ejemplificar algunos de los tantos juegos que pueden darse (en donde las flechas van llevando de patrón ventajoso a patrón ventajoso, hasta llegar al final del juego en donde el patrón ventajoso se convierte en ganador):

1-3-5-7 →1-3-5→ **1-3-2**
1-1-5-5 →1-5-5→ **5-5**
1-1-4-4 →1-1-4→ **1-1-1**
1-1-3-3 →1-1-3→ **1-1-1**
1-1-2-2 →1-2-2→ **2-2**
1-2-3 →1-2→ **1-0**
1-1-1 →1-1→ **1-0**
5-5 →5-4→ **4-4**
4-4 →4-3→ **3-3**
3-3 →3-2→ **2-2**
2-2 →2-0→ **1-0**

4.5. Patrones en ajedrez

Considérese un patrón, de acuerdo a Alexander [3], este consiste en una regla que contiene contexto, problema y solución y podemos pensar en el ajedrez en esos términos. Para hacer esto, podemos definir patrones que correspondan a las diferentes etapas del juego como son las aperturas (el inicio de la partida), el medio juego y los finales. Similarmente, podemos tener una sub–clasificación que tiene que ver con el tema: táctico o estratégico. En otras palabras, hay posiciones que pueden resolverse con una secuencia clara de jugadas, por acciones tácticas; mientras que otras posiciones se resuelven únicamente por factores estratégicos o posicionales. Por lo tanto, el contexto de cada patrón debe estar bien definido. Lo que importa en todo caso, es que claramente se vea la conexión de los elementos.

Podemos tener patrones muy complejos en ajedrez, por ejemplo, el llamado "principio de las dos debilidades", el cual se define de la siguiente manera: "uno de los jugadores tiene dos debilidades, una en cada lado del tablero. El

rival, a su vez, tiene más espacio en el centro, por el que ir de un lado a otro del tablero lo puede hacer de manera más rápida que el oponente. Tarde o temprano, el lado más fuerte puede atacar una debilidad seguido de ir a atacar la segunda debilidad y entonces, regresar a la primera. La mayoría de las veces, el oponente no tendrá tiempo de llegar para defender ambas debilidades. La expresión de este principio es simple, pero categorizarla en un patrón parece ser más complicado.

Los patrones tratan de diseño e interacción de los objetos, así como el de proveer una plataforma de comunicación con una solución reusable a los retos que se encuentran comúnmente[2]. Así pues, los patrones para el juego del ajedrez son una abstracción de propósito general para el problema de jugar el ajedrez de forma apropiada. Una buena colección de patrones puede reducir la complejidad y solucionar los problemas que se encuentran contidianamente en el ajedrez, haciendo que sean más fáciles de entender, por lo que podríamos tener una selección de soluciones probadas para trabajar con ellas.

Podemos definir algunos de los patrones más populares y conocidos en ajedrez: la "fórmula de Tarrasch", el doblete del caballo, la regla del cuadrado del peón, etcétera.

4.6. Algunos patrones populares en ajedrez

4.6.1. La *fórmula de Tarrasch*

Los finales más frecuentes –y los más difíciles de jugar bien– son los finales de torres. Aparecen cuando han desaparecido las otras piezas y sólo quedan torres y peones. Muchas veces, en este tipo de finales, hay un peón de más por parte de uno de los jugadores y en general ese peón es pasado, es decir, no hay un peón oponente en la misma columna que impida su avance. Hay muchos consejos y recomendaciones para ambos jugadores, tanto para quien tiene la ventaja como para el que se está defendiendo. El gran maestro polaco Siegbert Tarrasch (1862–1934), enunció una regla muy interesante para poder manejar este tipo de finales.

[**También conocido como**] - *Las torres deben colocarse detrás de los peones pasados* –ya sea el peón propio o el de su oponente.

[**Ejemplo**] - En partida **Mecking – Korchnoi**, 1974[3], las negras llegaron

[2]http://www.dofactory.com/net/design-patterns

[3]http://www.chessgames.com/perl/chessgame?gid=1082252

a la posición en donde se tiene un peón pasado en la columna “a”, en un final típico de torres y peones. Este es un ejemplo perfecto de la fórmula de Tarrasch (ver figura 4.4) en donde el jugador que se defiende debe aplicar dicha regla para salvar la partida (empatándola):

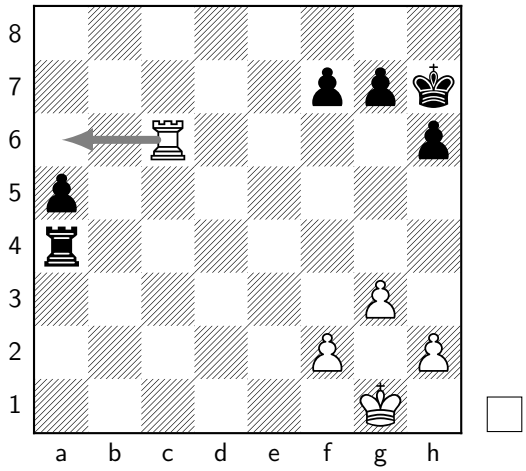

Figura 4.4: **Mecking–Korchnoi**, 1974.

Un ejemplo de la “fórmula de Tarrasch”. La torre blanca va detrás del peón pasado negro en la columna “a”.
[**Contexto**] - En los finales donde damas, alfiles y caballos ya se han cambiado, los peones (que no tienen peones enemigos enfrente de ellos) son elementos de fuerza en favor del jugador con ventaja. Es sabido que un peón que llega al final del tablero puede convertirse en una pieza más poderosa (a excepción del rey).
[**Problema**] -El jugador en desventaja debe detener el peón pasado del oponente.
[**Solución**] - Colocar una torre detrás del peón pasado. “En los finales de torres complicados, la regla más importante es la siguiente: La torre debe colocarse detrás del peón pasado, detrás del peón enemigo, ya sea para impedir que se promueva en una pieza más fuerte o en el caso del peón propio, para apoyar el avance [77]”.
[**Estructura**] - Los participantes más usuales en este tipo de posiciones son las torres y los peones. Frecuentemente uno de los jugadores tienen un peón pasado de más, lo que significa que no hay peón oponente en esa columna

que impida su avance.
[**Dinámica**] - La figura 4.5 muestra la maniobra por parte del lado que se defiende.

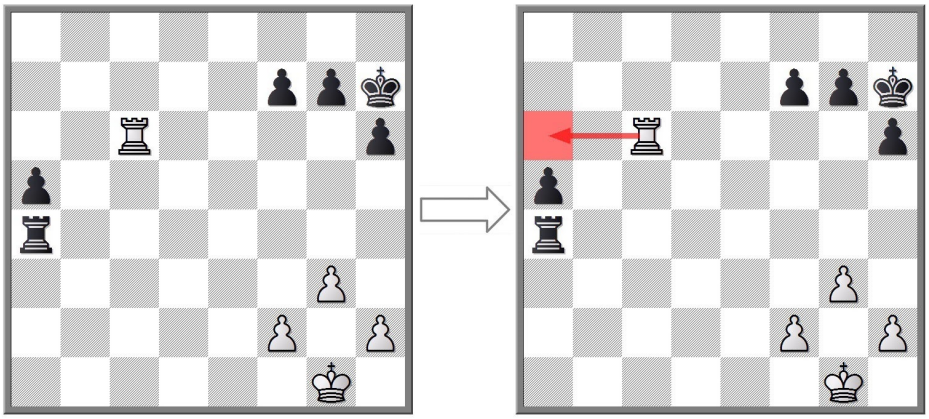

Figura 4.5: La "fórmula de Tarrasch" en acción

[**Variantes**] - La "fórmula de Tarrasch" tiene algunas excepciones (véase *Usos conocidos*). Por ejemplo, en la partida entre **Kharlov** – **Morozevich**, 1995[4] (ver figura 4.6), la jugada sugerida por la regla mencionada, **1...♖b7** solamente lleva al empate.

[4]http://www.chessgames.com/perl/chessgame?gid=1099966

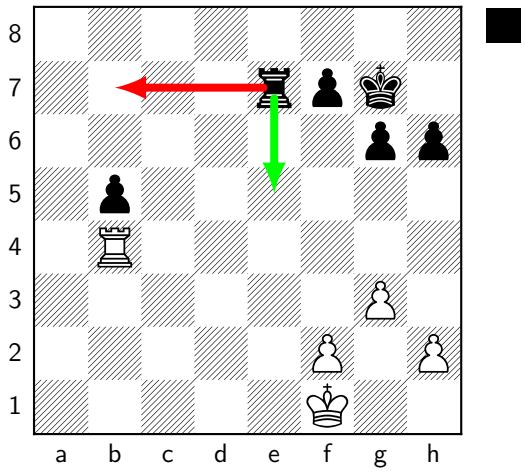

Figura 4.6: Aquí es equivocado usar la fórmula de Tarrasch.

Poner la torre detrás del peón no es suficiente para ganar (cuadro "b7") (fin de la flecha marcada en rojo), ya que el peón libre queda bloqueado y es posible acercar al rey blanco a la columna "b". La maniobra correcta es poner la torre negra en la casilla "e5" (marcada con verde).

[**Usos conocidos**] - La fórmula de Tarrasch es válida en general, pero tiene sus excepciones:

- Yuri Averbach [8] indica que la regla es correcta usualmente cuando ambas torres, la del jugador con ventaja y la del jugador que se defiende, están sobre el peón [63].
- Pero cuando el peón está bloqueado por el rey enemigo, la torre del mismo color está se coloca en mejor posición si lo defiende de lado [37].
- En el final de torre y peón contra torre, si el peón no ha pasado la cuarta fila, el mejor lugar para la torre es enfrente de su propio peón [34].
- En una anotación similar, Cecil Purdy indica que la torre está mejor detrás del peón si éste ha alcanzado al menos la quinta fila, [68] pues cada vez que se avanza el peón, la torre cobra más fuerza.

[**Consecuencias**] - Muchos finales de torres se dan con la ventaja mínima del peón pasado. El jugador con la ventaja debe colocar su torre detrás de su peón para apoyarlo. El defensor debería intentar hacer lo mismo. Poner una torre detrás del peón enemigo es una manera segura de empatar en una posición inferior. Muchos finales de torres se han salvado usando esta fórmula, es decir, han terminado en empate[5].

4.6.2. Eliminación de defensor

Los maestros de ajedrez, con mucha frecuencia, colocan sus fuerzas (piezas), de manera que la mayoría de las ocasiones se defienden mutuamente. Un truco táctico común en contra de esta idea estratégica es eliminar al defensor de una pieza (puede ser también un peón) de forma que al capturarse, ese jugador quede con una pieza indefensa que entonces puede capturarse. Esta es una manera de asegurarse ventaja material.

[**Ejemplo**] En la siguiente posición (ver figura 4.7), el alfil negro defiende a la torre en "f5". El procedimiento para ganar la torre es remover el defensor para así poder capturar la pieza defendida.

[5] *Los finales de una torre y peones son los que aparecen con más frecuencia en las partidas de ajedrez. Esto no significa que sea fácil jugarlos y pocos ajedrecistas los dominan. Son frecuentemente de una naturaleza muy difícil aunque aparentemente parezcan muy sencillos. La realidad es que pueden ser extremadamente intrincados.* - José Raúl Capablanca [8].

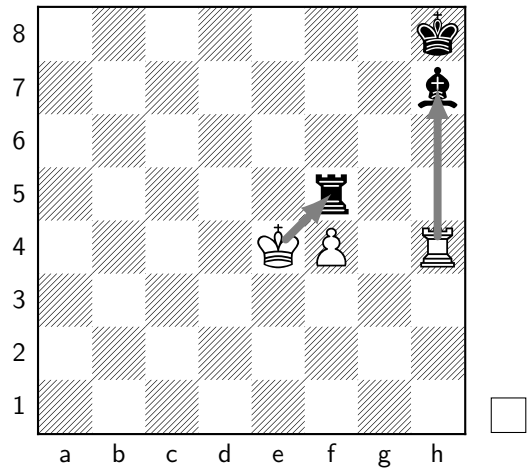

Figura 4.7: Eliminación del defensor.

[**Contexto**] -Piezas que están siendo atacadas y que son defendidas por otras piezas.
[**Problema**] - Uno de los jugadores está atacando una pieza que a su vez, defiende a otra pieza.
[**Solución**] - Quitar al defensor de la pieza que se ataca para capturar ésta en la siguiente jugada.
[**Estructura**] - Este es un patrón general en el cual muchas piezas pueden estar involucradas. Frecuentemente las piezas se defienden mutuamente por lo que eliminar al defensor es un procedimiento táctico común.

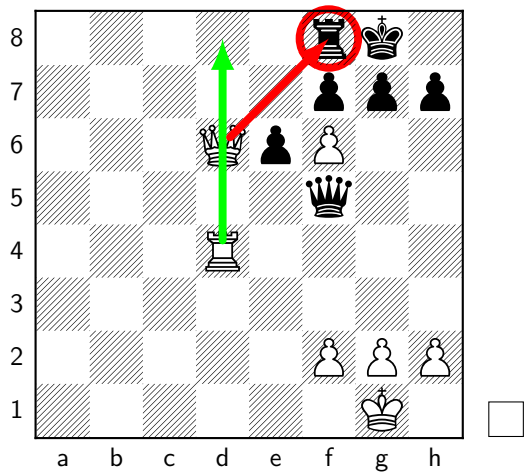

Figura 4.8: Un caso cuando el defensor es eliminado para dar mate en *d8*

.

[**Ejemplo resuelto**] (Ver figura 4.8). Las blancas buscan el mate eliminando el defensor (la torre en "f8") y dando mate con la torre. La secuencia es: **1** ♕**×f8+! ♔×f8** seguido de **2** ♖**d8#**

[**Variantes**] - Hay muchas variantes de este patrón, pero en principio es el mismo: eliminar el defensor para tomar la pieza defendida.
[**Usos conocidos**] - Esta es una maniobra muy común y hay muchísimos ejemplos en la práctica magistral de ajedrez.
[**Consecuencias**] - El aplicar este patrón lleva frecuentemente a hacerse de la ventaja o al menos tener más oportunidades de un mejor juego al ejecutarlo. Hay sin embargo un pero: esta maniobra en ocasiones sobresimplifica la posición de forma que a veces no hay suficiente material para ganar.

4.6.3. El regalo griego

Este tema táctico está inspirado en la leyenda de la Guerra de Troya, en donde los griegos, creando un caballo de madera enorme de regalo a los oponentes, pudieron entrar a la ciudad de Troya y ganar la guerra. En ajedrez, el jugador que está atacando algunas veces puede llegar a tener una configuración de piezas que pueden permitir iniciar un fuerte ataque, empezando

por el sacrificio de un alfil en contra de los peones que defienden al rey rival. La secuencia de movimientos del lado que ataca es muy simple de seguir y el defensor en muchas ocasiones no es capaz de defenderse adecuadamente de este sacrificio agresivo.

[**También conocido como**] - Sacrificio en *h7/h2*

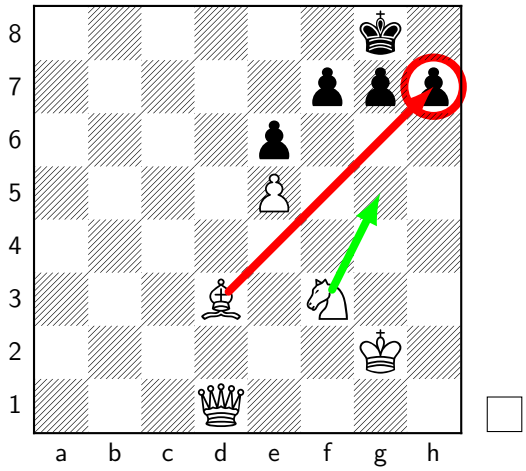

Figura 4.9: Ejemplo del sacrificio griego.

[**Ejemplo**] - La figura 4.9 tiene los elementos fundamentales para el *regalo griego*.
[**Contexto**] - Los elementos fundamentales son el alfil atacando "h7", la dama pudiéndose desplazar a "h5" y un caballo en "f3".
[**Problema**] - Destruir la fortaleza del rival para iniciar un ataque muy poderoso que lleve al triunfo.
[**Solución**] - En la posición del diagrama, **1 ♗×h7+** inicia el ataque. Después de **1...♔×h7 2 ♘g5+** y **3 ♕h5** se construye un muy fuerte ataque contra el desprotegido rey rival.
[**Estructura**] - Este patrón ocurre frecuentemente cuando el rey rival está enrocado corto y las piezas del atacante están en as posiciones para iniciar una fuerte ofensiva: Un alfil atacando el peón de "h7", un caballo que domina la casilla "g5" y una dama que puede trasladarse al flanco rey, donde se desarrolla el ataque.

[**Dinámica**] - La figura 4.10 muestra la secuencia de movimientos por hacer.

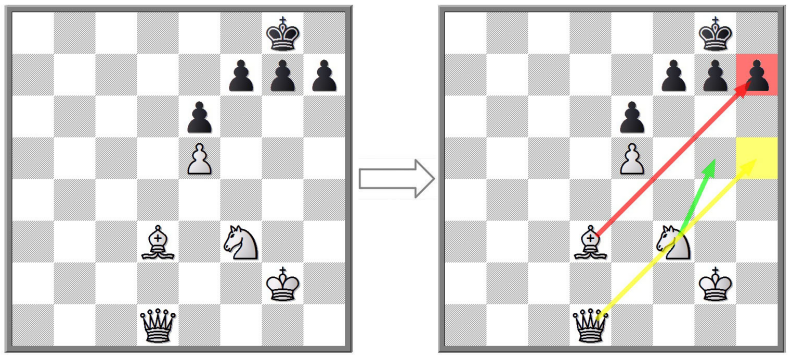

Figura 4.10: Dinámica del regalo griego: **1 ♗×h7+! ♔×h7 2 ♘g5+ ♔g8 3 ♕h5** con un fuerte ataque.

[**Ejemplo resuelto**] En la siguiente partida **Colle – O' Hanlon**, Niza 1930[6], se llegó a esta posición (ver figura 4.11). Aquí las blancas entregaron el alfil (el regalo griego) para montar un muy fuerte ataque.

[6]http://www.chessgames.com/perl/chessgame?gid=1316498

Figura 4.11: Dinámica del regalo griego: **1 ♗×h7+! ♔×h7**

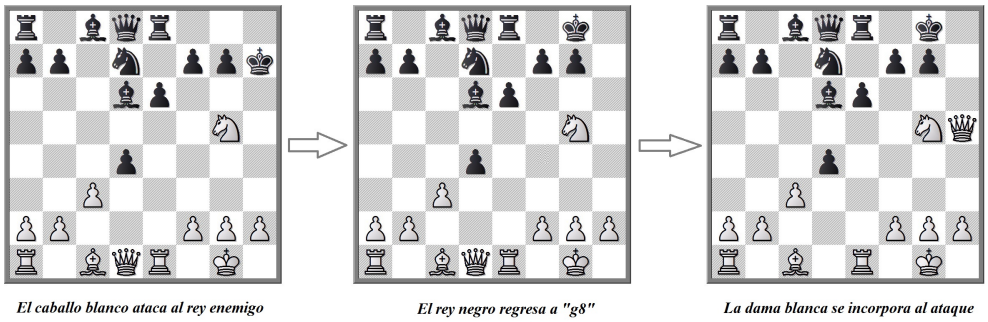

Figura 4.12: Dinámica del regalo griego (continuación): **2 ♘g5+ ♔g8 3 ♕h5** con un ataque ganador.

[**Variantes**] - Hay variaciones del regalo griego, pero en este caso, la posición debe tener los elementos fundamentales para tener éxito en el ataque.

[**Usos conocidos**] - Hay muchas partidas de la práctica magistral en donde se observa ese patrón. Una de las más impresionantes es Wells, P. – Dumitrache, Dragos, Balatonbereny 1997. **1 d4 d5 2 c4 e6 3 ♘c3 c6 4 ♘f3 ♘f6 5 e3 ♘bd7 6 ♗d3 ♗d6 7 O-O O-O 8 e4 ♘×e4 9 ♘×e4 d×e4 10 ♗×e4 ♖e8 11 ♖e1 c5 12 ♗×h7+ ♔×h7 13 ♘g5+ ♔g6 14 g4 ♘f8 15 ♕d3+ f5 16 ♕h3 ♔f6 17 ♕h5 f4 18 ♖e5 ♕d7 19 ♗×f4 ♗×e5 20 ♗×e5+ ♔e7 21 ♘f7 ♕a4 22 ♗d6+ ♔d7 23 ♕×c5 e5 24 d5 ♘e6 25 ♘×e5+ ♔d8 26 ♗c7+** 1-0.

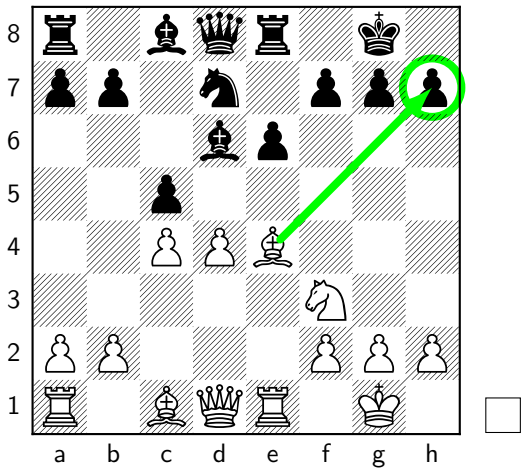

Figura 4.13: El regalo griego en la partida **Wells P.** – **Dumitrache D**, Balatonbereny, 1997. **12 ♗×h7+**

[**Consecuencias**] - Generalmente este patrón es muy conocido y funcional. Pero es de hacer notar que todos los elementos deben estar en la posición. Por ejemplo, en la partida **Salgado, I. - Svidler, P.**, Gibraltar 2015[7]. Salgado no tenía todos los elementos del patrón para construir un fuerte ataque, por lo que las negras se defendieron con éxito del sacrificio griego, aunque esto no es lo usual.

4.6.4. Regla del cuadrado del peón

"Los peones son el alma del ajedrez"[8]. En muchos finales de peones (donde solamente hay reyes y peones) el cálculo concreto de variantes (de los movimientos), es la preocupación principal de los ajedrecistas, porque tienen que ser muy precisos. Cualquier error puede significar la derrota. La regla del cuadrado es un atajo muy ingenioso que elimina la necesidad de calcular exactamente el futuro de los movimientos. Esta maniobra es común aplicarla

[7]http://www.chessgames.com/perl/chessgame?gid=1783557

[8]Con esta cita, Danican André Philidor (1726-1795), definió el elemento fundamental de la estrategia ajedrecística. **Analyse du jeu des échecs**, François Danican Philidor, *Chez P. Elmsley, 1777.*

en los finales de peones.

[**Ejemplo**] En la siguiente posición (figura 4.14):

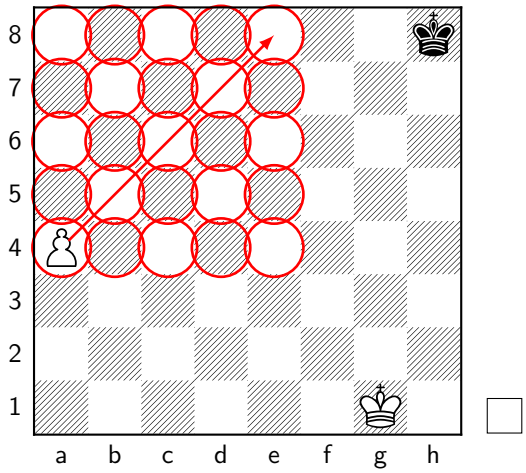

Figura 4.14: Si el rey negro se queda fuera del cuadrado del peón (en círculos rojos), éste podrá llegar a coronar y convertirse en una pieza más fuerte. En caso contrario, el peón puede ser capturado por el rey enemigo.

[**Contexto**] -Finales de reyes y peones, peones pasados y promociones.
[**Problema**] - Cada peón tiene una casilla de coronación. El rey enemigo desea impedir eso.
[**Solución**] - Si el rey está dentro del cuadrado definido por el peón, entonces lo puede atrapar antes de que corone. Si el rey está fuera de esa área, no puede alcanzarlo. El cuadrado se hace trazando una diagonal desde donde está el peón hasta la orilla del jugador rival (vea la figura 4.14).
[**Estructura**] - Este patrón ocurre en la mayoría de los finales de peones en donde ya las otras piezas han sido intercambiadas. Solamente involucra a reyes y peones[9].

[9]Cabe señalar que los finales de caballos y peones se manejan bajo los mismo principios que los finales de reyes y peones solos, de acuerdo con el excampeón mundial Mijaíl Botvinnik (**Botvinnik: One Hundred Selected Games**, Mijaíl Botvinnik, *Dover*, 1960.)

[**Dinámica**] En lugar de contar movimientos para ver si se puede atrapar el peón pasado rival, al jugador le basta checar si el rey está en el cuadrado del peón. Este es un atajo a los cálculos en el tablero de ajedrez.
[**Ejemplo resuelto**] - Durante el campeonato mundial de computadoras de la ACM, en 1984, en la última ronda, la 4, en la partida de **NuChess** (2 puntos de 3) contra **Cray Blitz** (3 puntos de 3), se llega a la siguiente posición (figura 4.15) [10] [11]:

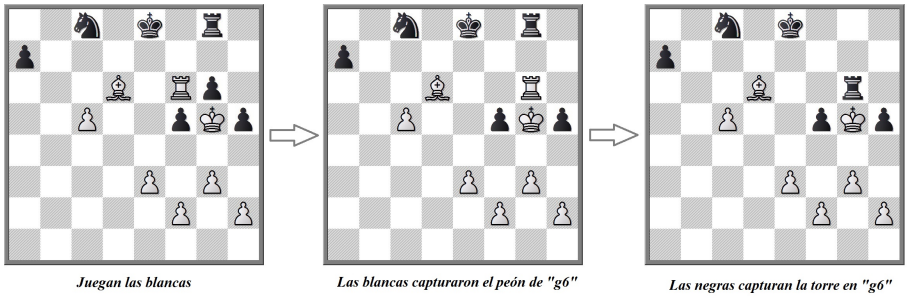

Figura 4.15: Secuencia de la partida **NuChess–Cray**, 1984. Después de **45 ♖×g6?? ♖×g6+**

[**Variantes**] - Hay muchas variantes de este patrón, pero básicamente se encuentra y se aplica en los finales de reyes y peones.

[10] **1 c4 e5 2 ♘c3 ♗b4 3 a3 ♗×c3 4 d×c3 ♘e7 5 g3 d5 6 c×d5 ♕×d5 7 ♕×d5 ♘×d5 8 ♗g2 ♘b6 9 a4 O-O 10 a5 ♘c4 11 ♖a4 ♘d6 12 a6 ♘d7 13 ♗e3 ♘b6 14 ♖h4 ♖d8 15 a×b7 ♗×b7 16 ♗×b7 ♘×b7 17 ♘f3 ♖d5 18 c4 ♖a5 19 O-O ♖a2 20 ♖d1 ♖×b2 21 c5 ♘c8 22 ♖d7 f6 23 ♖g4 g6 24 ♖h4 h5 25 ♖×c7 ♘d8 26 ♖a4 ♖b7 27 ♖×b7 ♘×b7 28 ♖a6 ♔f7 29 ♘d2 ♘d8 30 ♘e4 f5 31 ♘g5+ ♔g7 32 ♘f3 ♘f7 33 ♘×e5 ♘×e5 34 ♗d4 ♔g8 35 ♗×e5 ♘e7 36 e3 ♔f7 37 ♖f6+ ♔g8 38 ♔g2 ♖c8 39 ♔f3 ♖e8 40 ♖a6 ♖a8 41 ♔f4 ♔f7 42 ♔g5 ♖g8 43 ♖f6+ ♔e8 44 ♗d6 ♘c8 45 ♖×g6 ♖×g6+ 46 ♔×g6 ♘×d6 47 c×d6 a5 48 g4 h×g4 49 ♔×f5 a4 50 e4 a3 51 ♔×g4 a2 52 e5 a1♕ 53 f4 ♕g1+ 54 ♔f5 ♕×h2 55 e6 ♕c2+ 0-1**. Aparentemente NuChess podía ganar la partida, logrando empatar el primer puesto, pero en este momento juega **45 ♖×g6??** y esta jugada convierte una posición ganadora en perdedora, logrando que Cray Blitz se hiciese del primer puesto. Las blancas ganan un segundo peón, pero el material a favor no es suficiente. En este caso el peón de "a" de las negras es inalcanzable. William Blanchard, del equipo de NuChess, indica que su programa no incluía el principio del cuadrado del peón por lo que su búsqueda de jugadas no llegaba a ser lo suficientemente profundo para calcular que las blancas no podrían alcanzar al peón. Hyatt en cambio dijo que Cray sabía de este principio y de su relevancia pero no esperaba que fuese a ser usado por su programa en ninguna circunstancia.

[11] https://chessprogramming.wikispaces.com/Rule+of+the+Square

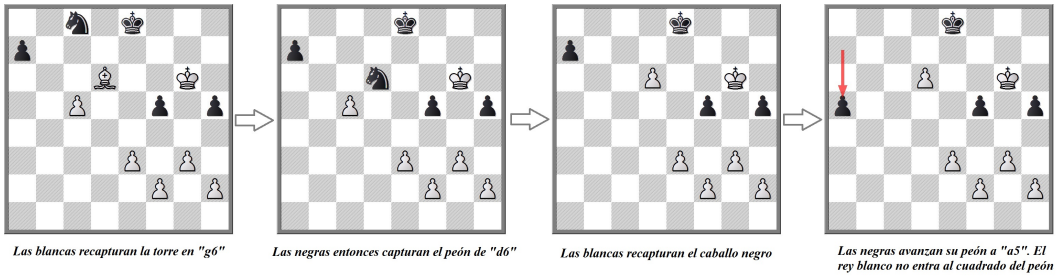

Figura 4.16: **46 ♔×g6 ♘×d6 47 c×d6 a5!** y las blancas no pueden atrapar al peón negro de la columna "a".

[**Usos conocidos**] - Esta es una maniobra común y hay muchos ejemplos de la práctica magistral.
[**Consecuencias**] - El aplicar este patrón puede ahorrarnos el tener que hacer un cálculo de las jugadas y respuestas en la partida. En algunos casos este patrón se usa en los finales de peones y caballos, aunque no con mucha frecuencia. El beneficio principal es que se ahorra el tener que calcular jugadas adelante.

4.6.5. Ataque doble del caballo

Hay un incontable número de patrones similares, en donde en muchos casos las posiciones pueden ser tratadas de la misma manera, aunque superficialmente parezcan ser posiciones diferentes. En el lenguaje de patrones podemos hacer esta generalización. La siguiente posición (figura 4.17) muestra el doble ataque del caballo:

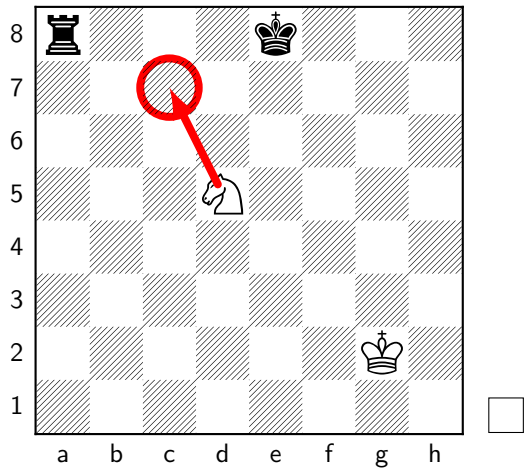

Figura 4.17: Un doble ataque del caballo típico.

Pero esta posición podría aparecer en otra parte del tablero: a través de los patrones, podemos deshacernos de la dependencia en las situaciones particulares, creando una generalización como muestra la figura 4.18:

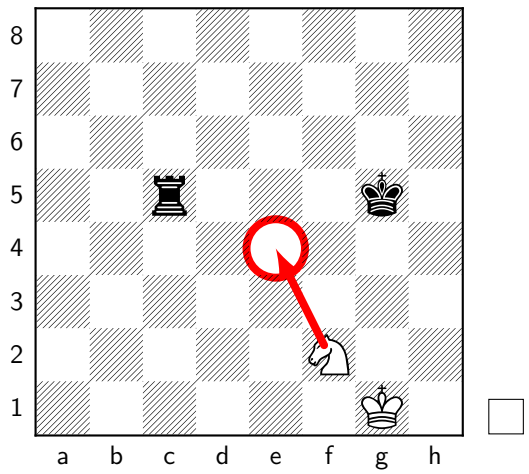

Figura 4.18: Una posición diferente con el mismo patrón del jaque doble del caballo.

[**También conocido como**] - *Doblete* del caballo.
[**Contexto**] - Los elementos muestran a un caballo con la amenaza de un ataque a dos piezas enemigas al mismo tiempo.
[**Problema**] - Eliminar la fuerza de las piezas rivales a través de un ataque doble de caballo.
[**Solución**] - Ejecutar el doble ataque para ganar una pieza del oponente. En la posición del diagrama (figura 4.19), **1 ♘e4+** gana la torre de *c5*.
[**Estructura**] - Este patrón siempre involucra el uso de un caballo y dos piezas de fuerza significativa del rival.
[**Dinámica**] Cada ataque doble de caballo debe ejecutarse en secuencia: primero, encontrar la casilla del doble jaque, checando que esa casilla no esté defendida y entonces sí, ejecutar el ataque doble contra las piezas pesadas del oponente.

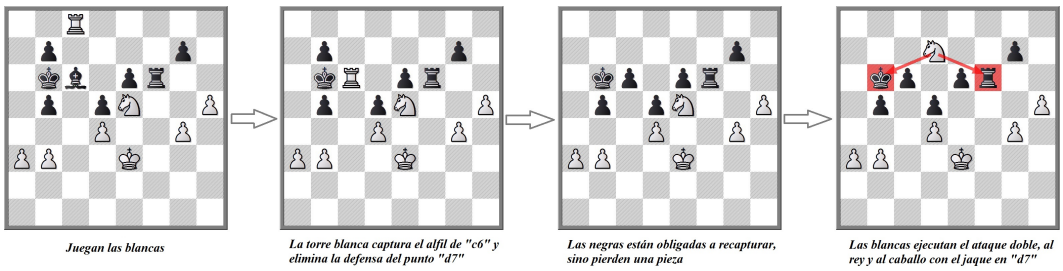

Figura 4.19: La secuencia completa en la ejecución de un doblete de caballo.

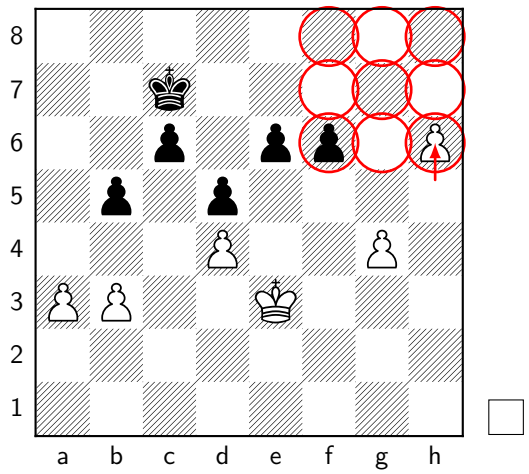

Figura 4.20: Dinámica del doblete de caballo en la partida **Larsen–López M**, Simultáneas (*circa*) 1971. Este es el segundo patrón, que aparece después del doblete, el del *cuadrado del peón* .

[**Ejemplo resuelto**] - El diagrama(ver figura 4.19), muestra la secuencia a realizar[12].
[**Variantes**] - Hay muchas variaciones de los dobletes y es un tema táctico muy común incluso en las partidas de los principiantes.
[**Usos conocidos**] - Hay incontables partidas con este patrón. Algunas de ellas notables[13].

[12]**Larsen, Bent–López Michelone, M.**, *circa* 1973. **1 d4 e6 2 e4 d5 3 ♘d2 ♘f6 4 e5 ♘e4 5 ♗d3 ♘×d2 6 ♗×d2 c5 7 c3 ♘c6 8 ♘f3 ♕b6 9 ♕a4 ♗d7 10 ♕b3 c×d4 11 ♕×b6 a×b6 12 c×d4 ♗b4 13 ♔e2 ♗×d2 14 ♔×d2 ♘b4 15 a3 ♘×d3 16 ♔×d3 ♗b5+ 17 ♔d2 ♖c8 18 ♖hc1 ♔e7 19 ♘e1 ♗a4 20 ♘d3 f6 21 ♖×c8 ♖×c8 22 ♖c1 ♗c6 23 h4 h5 24 ♘f4 f×e5 25 ♘g6+ ♔d6 26 ♘×e5 ♖f8 27 ♔e3 ♖f6 28 b3 b5 29 f3 ♔c7 30 g4 h×g4 31 f×g4 ♔b6 32 h5? ♗e8 33 ♖c8 ♗c6 34 ♖×c6+! b×c6 35 ♘d7+ ♔c7 36 ♘×f6 ♔d8 37 g5 ♔e7 38 ♘g4 ♔f7 39 ♘e5+ 1-0**. La posición final muestra de hecho otro patrón: *la del cuadrado del peón*. En este caso, las negras no pueden detener el avance del peón blanco pasado en la columna "h"

[13]Karjakin, Sergey – Anand, Viswanathan, Moscow 2016, **1 ♘f3 d5 2 e3 ♘f6 3 c4 e6 4 b3 ♗e7 5 ♗b2 O-O 6 ♘c3 c5 7 c×d5 ♘×d5 8 ♕c2 ♘c6 9 h4 b6 10 a3 f5 11 ♗b5 ♗b7 12 ♘×d5 e×d5 13 d4 ♖c8 14 d×c5 b×c5 15 O-O ♗f6 16 ♖fd1 ♘e7 17 ♗×f6 ♖×f6 18 g3 ♗a6 19 ♗×a6 ♖×a6 20 ♕c3 ♖b6 21 ♖ac1 ♕d6 22 ♘e5 ♖b7 23 ♘d3 c4 24 b×c4 ♖×c4 25 ♕e5 ♕×e5 26 ♘×e5 ♖×c1 27 ♖×c1 g6 28 ♖c5 ♔g7 29 ♖a5 ♔f6 30 ♘d3 ♖c7 31 ♖a6+ ♔g7 32 ♘f4 ♖d7 33 ♔f1 ♘g8 34 ♘e6+ ♔f7**

[**Consecuencias**] - La aplicación de este patrón trae consigo ventaja material en general.

4.6.6. El patrón La clavada

Se dice que una pieza está "clavada" cuando no se puede mover. Por ejemplo, esta situación pasa cuando un alfil ataca un caballo, el cual no puede quitarse porque en esa misma diagonal del ataque se encuentra una pieza de mayor valor, en general el rey o la dama.

[**Ejemplo**] - La siguiente posición hipotética marca los elementos fundamentales de *la clavada*. En la posición del diagrama (figura 4.21), el caballo de "g6" no puede moverse porque si lo hiciese, el rey negro quedaría en jaque (produciéndose una jugada ilegal). A esto se le llama una *clavada absoluta*. El caballo simplemente está como "clavado" al tablero. No se puede mover. En cambio, el alfil de b4 negro está clavado por la torre blanca de b1, pero en este caso sí podría moverse, ya que no es el rey, sino la dama negra, quien se encuentra atrás. A esto se le llama una clavada *relativa*. A veces creemos que una pieza no puede moverse por estar, precisamente, clavada y de pronto ésta se mueve, mostrándonos que por no ser una *clavada absoluta*, es decir, no está atrás el rey enemigo, finalmente sí se puede mover. Ahora bien, en esta misma posición las blancas tienen clavada su dama (y es una clavada absoluta pues el rey está atrás de la dama blanca en diagonal, pero atrás).

35 ♘d4 ♘e7 36 ♘b5 ♘c8 37 a4 ♖b7 38 ♖c6 ♘e7 39 ♖a6 ♘c8 40 ♖c6 ♘e7 41 ♖d6 ♖b6 42 ♖d7 a6 43 ♘c3 1-0

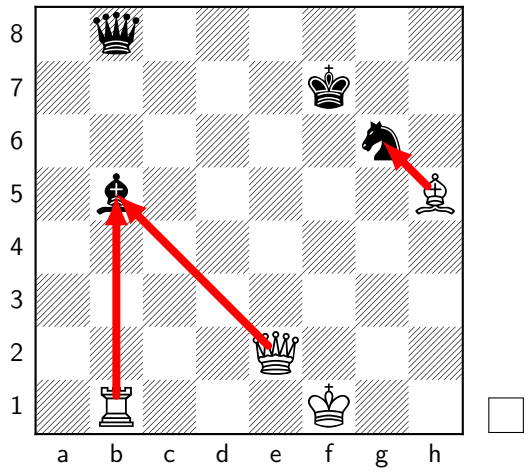

Figura 4.21: Un juego de clavadas.

[**Contexto**] - El elemento común es una pieza de un jugador que impide mover a una pieza del oponente por hallarse atrás una pieza de mayor valor.
[**Problema**] - La clavada impide que el rival pueda mover una pieza, haciendo que las piezas o casillas que defiende sean una ilusión.
[**Solución**] - Eliminar la clavada quitando la pieza de mayor valor que se encuentra atrás o bloqueando la acción del atacante.
[**Estructura**] -
[**Dinámica**] - El diagrama (figura 4.22) muestra la secuencia a realizar.
La última jugada del blanco **1 ♗f6??** (diagrama de la izquierda en la figura 4.22), resulta un error. La torre negra ha sido clavada de manera absoluta por el alfil. Además, la dama blanca amenaza con capturar la torre enemiga. Kasparov, ha visto más lejos y ha encontrado que la dama se encuentra en una clavada absoluta a su vez. No obstante, la torre negra no puede capturar a la dama por estar, repetimos, a su vez clavada de manera absoluta por el alfil. Después de **1. . . ♕d1+!** La casilla d1 realmente no está defendida por la dama. Aquí Ehlvest se rinde pues después de **2 ♔g2 ♕×g4+ 3 ♔h1 ♕d1#**. [**Variantes**] - Las clavadas se dan en todas las fases de las partidas. Es muy común clavar un caballo enemigo en f3/f6 o c3/c6.
[**Usos conocidos**] - Hay infinidad de partidas con este patrón. Un ejemplo ya famoso se dio en la partida **Kotov, A.–Botvinnik, M**, URSS 1939[14] (véase

[14]http://www.chessgames.com/perl/chessgame?gid=1031990

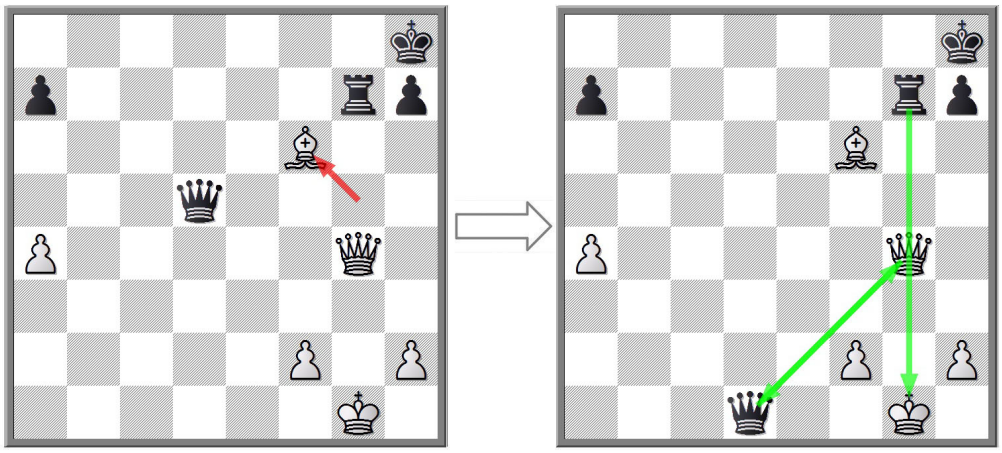

Figura 4.22: **Ehlvest** – **Kasparov**, Moscú, 1977. Un juego de clavadas y contra–clavadas.

figura 4.23) en donde Botvinnik remata de la siguiente manera: **37. . . ♕×g2+! 38 ♕×g2 ♖×e2** 0-1.
[**Consecuencias**] - El tema táctico de la clavada permite muchas veces la ganancia de espacio o incluso de material.

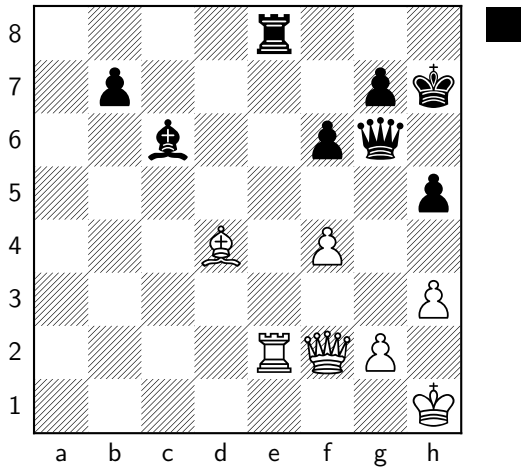

Figura 4.23: **Kotov** – **Botvinnik**, URSS 1939. Una clavada absoluta termina con la resistencia del blanco.

4.6.7. La Distracción

La *distracción* ocurre cuando una pieza oponente tiene que dejar su rol defensivo de una pieza propia o una casilla crítica para ocuparse de otras labores.
[**También conocido como**] - Diversión, deflexión o pieza sobrecargada.
[**Ejemplo**] - En **Moiseenko – Potkin**, Estambul 2003 (Figura 4.24), juegan las negras y a través de la deflexión o distracción, las negras se hacen de la ventaja ganadora: **19...♖ac8+ 20 ♔b1 ♗e4+ 21 ♔a1 ♖c1+!** (distrae a la torre de la defensa de la dama blanca en d6) **22 ♖×c1 ♕×d6** y las negras ganaron en la jugada 31.

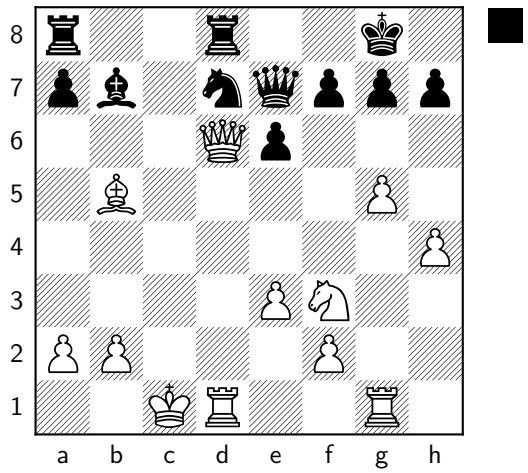

Figura 4.24: **Moiseenko – Potkin**, Estambul 2003. La torre blanca de "d1" está obligada a cubrir a la dama en "d6" y al mismo tiempo la casilla "c1".

[**Contexto**] - Cuando una pieza tiene más de una tarea defensiva.
[**Problema**] - En la práctica del ajedrez, los jugadores tienden a tener todas sus piezas bien defendidas, pero pueden ocurrir casos donde una pieza tiene que asumir dos o más defensas a la vez, y esto da el tema táctico de la distracción, haciendo que la pieza defensora tome acciones sobre el ataque de una de sus piezas y en ese momento deja desprotegida una casilla importante u otra pieza y el rival se puede hacer de la ventaja.
[**Solución**] - Distraer la labor del defensor para que deje desprotegida una

pieza o casilla importante.
[**Estructura**] - Este es un patrón general en el cual pueden estar involucradas muchas piezas. Con frecuencia tenemos piezas que defienden a otras piezas y cuando alguna de ellas hace más de una labor defensiva, es posible que surja el tema de la distracción.
[**Dinámica**] - El siguiente diagrama (figura 4.25) muestra a un alfil negro defendiendo dos casillas críticas ("d8" u "f8"). Juegan las blancas y ganan. La maniobra ganadora es distraer al alfil defensor de la protección de la casilla "f8": **1 ♖d8+! ♗×d8 2 ♖f8#**.

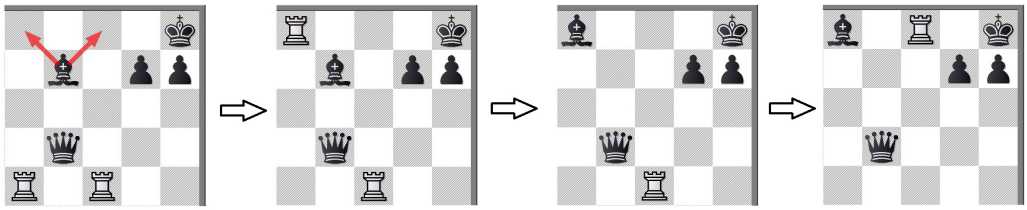

Figura 4.25: El alfil de "e7 tiene la misión de defender dos casillas críticas: "d8" y"f8".

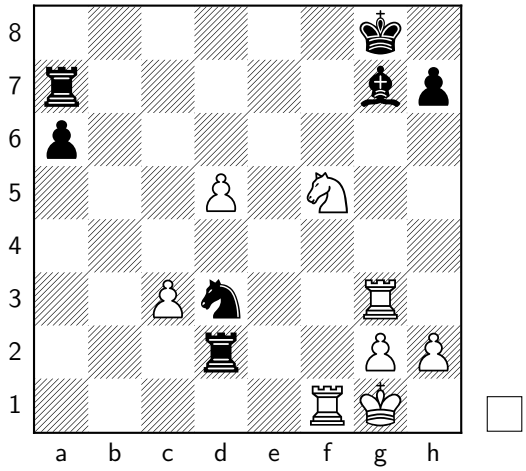

Figura 4.26: **Vitolinsh – Gadiarov**, Riga 1979, las blancas juegan y ganan.

[**Ejemplo resuelto**] - En la partida **Vitolinsh – Gadiarov**, Riga 1979, (ver figura 4.26), las blancas juegan y ganan. La distracción de la defensa ocurre en las casillas "f8" y "g8": **35 ♘h6+ ♔h8 36 ♖f8+ ♗×f8 37 ♖g8#** 1-0.
[**Variantes**] - El tema de la desviación es muy común y puede hallarse en muchísimas combinaciones ilustrativas del ajedrez magistral. Es uno de los patrones más conocidos.
[**Usos conocidos**] - La distracción es un tema recurrente en muchísimas partidas.
[**Consecuencias**] - La aplicación de este patrón en general lleva a que el jugador que lo ejecuta se haga de ventaja material decisiva.

4.6.8. Rey ahogado

El *rey ahogado* es una situación que se produce cuando el jugador de quien es el turno no tiene jugadas legales para realizar y su rey no se encuentra en estado de jaque. Es decir, el rey no puede moverse a otras casillas porque quedaría en posición de jaque o porque están ocupadas por piezas propias o piezas ajenas que están defendidas, y además el jugador no tiene otras piezas que puedan moverse o capturar a piezas adversarias. Esta situación conduce al empate.
[**También conocido como**] - Tablas por rey ahogado, mate ahogado.
[**Ejemplo**] - En **Evans – Reshevsky**, Campeonato de Estados Unidos 1963 (Figura 4.27)[15], juegan las blancas, las cuales están perdidas. Tienen material de menos y su rey está a punto de recibir mate. Evans encuentra el recurso de tablas por rey ahogado: **49 ♕g8+! ♔×g8 50 ♖×g7+** y la torre no puede ser capturada por el rey o la dama del rival, porque entonces el rey blanco queda ahogado. Se acuerda el empate.

[15] http://www.chessgames.com/perl/chessgame?gid=1252040

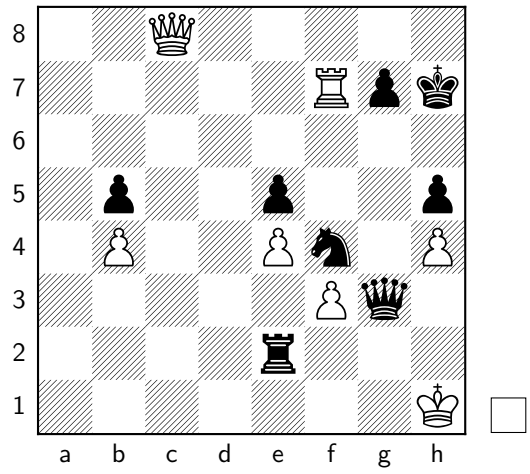

Figura 4.27: **Evans – Reshevsky**, Campeonato de Estados Unidos 1963. Juegan las blancas, las cuales se salvan de una derrota inminente gracias al recurso del rey ahogado.

[**Contexto**] - Es un último recurso del jugador en desventaja, buscando el empate. Deben de cumplirse dos condiciones: 1. que el rey del jugador en desventaja no tenga movimientos legales y 2. que sus otras piezas no puedan moverse o puedan ser sacrificadas para hallar el empate de esta manera.
[**Problema**] - Poder salvar una partida en donde se tiene desventaja buscando el recurso del rey ahogado, que no siempre está presente, pero que puede hallarse en posiciones desesperadas.
[**Solución**] - En muchas posiciones el jugador en desventaja, al borde de la derrota, encuentra un procedimiento en donde el rival es obligado a capturar las piezas enemigas hasta dejar sin movimientos al jugador que busca salvar la partida.
[**Estructura**] -
[**Dinámica**] - El siguiente diagrama (figura 4.28) muestra la posición crítica de la partida **Pilnick, C. – Reshevsky**[16], quien en una posición de ventaja ganadora se esfuma al encontrar su rival el recurso del ahogado. El jugador en desventaja debe tener presente que su rey no tiene movimientos y que sus otras piezas, o no se pueden mover o bien pueden entregarse para llegar a

[16]http://www.chessgames.com/perl/chessgame?gid=1441025

una posición de rey ahogado, como en este caso, que después de **93 ♕f2!!**, las negras tienen que capturar la dama blanca y aceptar el rey ahogado.

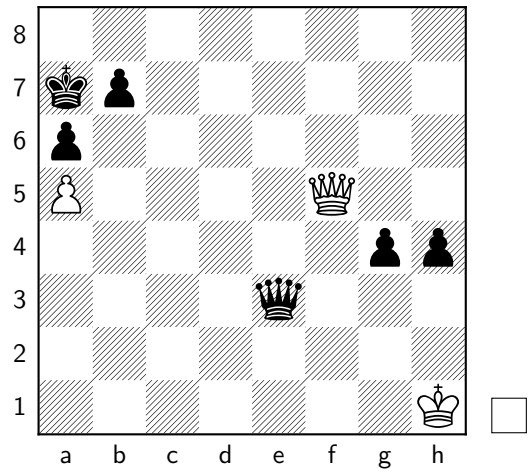

Figura 4.28: **Pilnick** – **Reshevky**, Campeonato de Estados Unidos, 1942. Las blancas usan su último recurso para llegar a una posición de rey ahogado. Cabe decir que esta "ceguera" de Reshevsky, uno de los jugadores más fuertes del mundo, lo persigue cada veinte años.

[**Ejemplo resuelto**] - En **Kasparov – McDonald, N.**, Gran Bretaña 1986 (figura 4.29)[17], el mejor jugador del mundo se tiene que conformar con el empate, después del recurso de rey ahogado que hallan las negras en una posición perdida: **54...♖×g3+!! 55 ♔×g3 ♕e5+!!** y se acuerda el empate porque las blancas tienen que aceptar el sacrificio de la dama negra, logrando el recurso del rey ahogado.

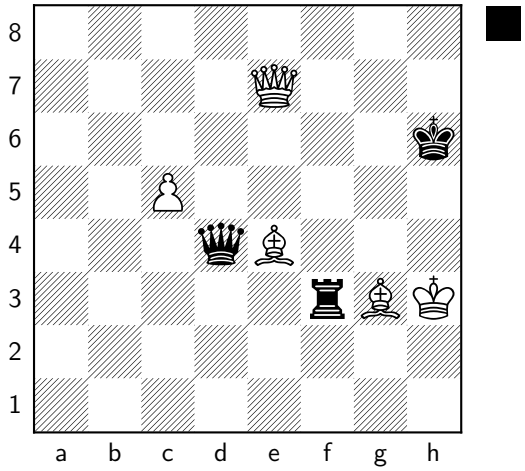

Figura 4.29: **Kasparov – McDonald, N**, juegan las negras y empatan.

[**Variantes**] - El tema del rey ahogado no es muy frecuente y en general ocurre al final de las partidas en donde hay menos material en el tablero. En la mayoría de los casos, el tema está relativamente escondido y el jugador que tiene ventaja subestima los recursos defensivos.

[17] http://www.chessgames.com/perl/chessgame?gid=1070210

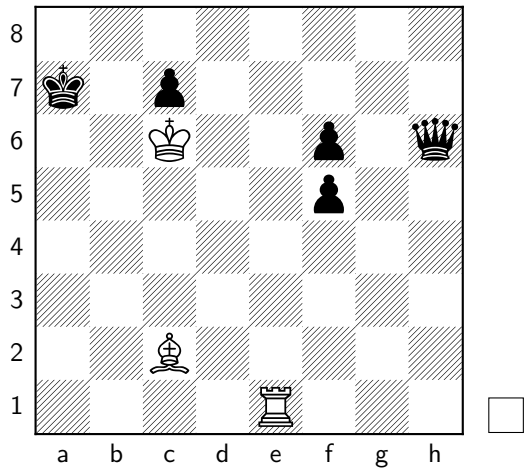

Figura 4.30: Estudio de Kubbel. Juegan las blancas y empatan.

[**Usos conocidos**] - El tema se ha utilizado frecuentemente en problemas y estudios de ajedrez compuestos, en donde el recurso del rey ahogado es parte de todo el problema. Por ejemplo, en el diagrama 4.30 se presenta un estudio de Kubbel, en donde el enunciado dice: *Juegan blancas y empatan.* La solución es la siguiente: **1 ♖a1+ ♔b8 2 ♖b1+ ♔c8 3 ♗×f5+!! ♕×f5 4 ♖b8+!! ♔×b8** y rey ahogado.
[**Consecuencias**] - La aplicación de este patrón en general ilustra los recursos defensivos del jugador en desventaja.

4.6.9. Jugada intermedia

La *jugada intermedia*, ocurre cuando un jugador, en vez de jugar el movimiento esperado (comúnmente una recaptura de una pieza que el oponente acaba de capturar) realiza antes otro movimiento, descubriendo una amenaza intermedia que el oponente tiene que responder, y una vez hecho esto, juega el movimiento esperado.
[**También conocido como**] - *zwischenzug* (palabra alemana que significa "jugada intermedia".

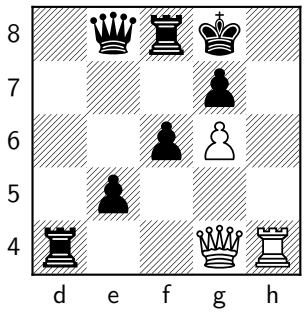

Figura 4.31: El tema de la jugada intermedia

[**Ejemplo**] - En la posición de la figura 4.31 se ejemplifica la jugada intermedia. Las blancas buscan dar mate colocando su dama en "h5" seguido de "h7". Sin embargo, no tienen tiempo de jugar directamente **1 ♕h5** por **1...♖×h5** y se desvanece el ataque. Por ende, realizan la jugada intermedia: **1 ♖h8+!** las negras están obligadas a capturar la torre sacrificada. **1...♖×h8** y entonces las blancas pueden proseguir con su plan: **2 ♕h5+** y **3 ♕h7#**.
[**Contexto**] - En muchas posiciones, la realización de una jugada intermedia, que obliga al rival a responder a ese movimiento, permite al jugador que lo ejecuta lograr la ventaja.
[**Problema**] - Hay ocasiones en el que la jugada intermedia significa la ganancia de un tiempo, de un movimiento más, para hacerse de la ventaja, en general cuando se está atacando al rey enemigo.
[**Solución**] - La jugada intermedia en general logra una ventaja que muchas veces se vuelve definitiva.
[**Estructura**] -

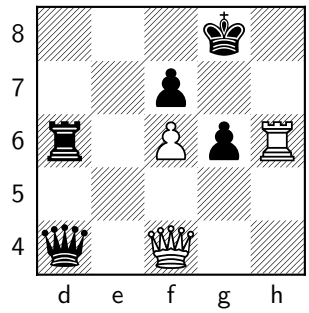

Figura 4.32: El tema de la jugada intermedia

[**Dinámica**] - En general, las jugadas intermedias promueven la ganancia de tiempos para continuar el ataque. Por ejemplo, en la siguiente posición (figura 4.32), las blancas tienen un fuerte ataque que el rival quiere contrarrestar cambiando damas. Pero ejecutando una jugada intermedia, las blancas logran un ataque ganador: **1 ♖h8+ ♔×h8 2 ♕h6+ ZZy 3 ♕g7#**
[**Ejemplo resuelto**] - En **Tartakower – Capablanca**, Nueva York 1924, (ver figura 4.33)[18], la última jugada del blanco fue **9 ♗×b8**, pensando que ganaba pieza después de **9...♖×b8** y **10 ♕a4+**, capturando el alfil indefenso, pero Capablanca, el tercer campeón del mundo encuentra una jugada intermedia que le da la ventaja: **9...♘d5!** y las blancas tienen que ocuparse de defenderse de la amenaza **10...♘e3+** doble. Capablanca ganaría 20 jugadas después[19].

[18]http://www.chessgames.com/perl/chessgame?gid=1076242

[19]irving Chernev, prolífico autor de libros la bautiza como "la Partida del Zwischenzug Inmortal"

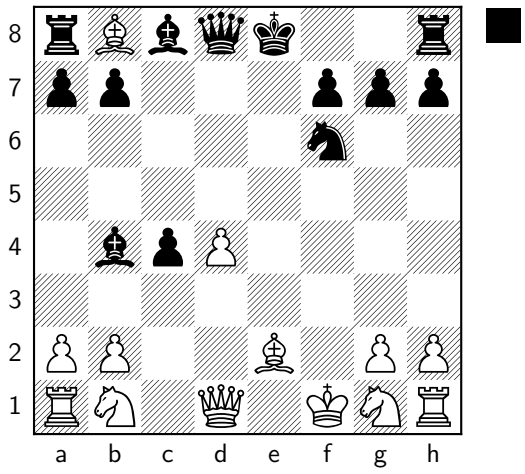

Figura 4.33: **Tartakower – Capablanca**, Nueva York 1924, juegan las negras y ejecutan una jugada intermedia que le da ventaja inmediatamente.

[**Variantes**] - Las jugadas intermedias son recurrentes, sobre todo en la práctica del ajedrez de elite, en donde en muchísimas ocasiones una jugada inesperada, intermedia, le da ventaja inmediata al jugador que la ejecuta.
[**Usos conocidos**] - El tema se ha utilizado en problemas y estudios de ajedrez compuestos, en donde en ocasiones la jugada intermedia es incluso muy sutil, pero que en profundidad logra la ventaja al jugador que ejecuta ese movimiento.
[**Consecuencias**] - La aplicación de este patrón ilustra que en ajedrez muchas combinaciones de jugadas pueden ejecutar movimientos de zwischenzug y no necesariamente seguir secuencias de movimientos lineales.

4.6.10. Destrucción de la estructura de peones

La *destrucción de la estructura de peones* hace que un jugador –en general– sacrifique material para debilitar la estructura de los peones que, por ejemplo, defienden al rey. Es un recurso del ataque que muchas veces tiene éxito, sobre todo cuando el jugador con la ventaja tiene un fuerte ataque contra el monarca rival.

[**También conocido como**] - *Destrucción de la estructura defensiva.*
[**Ejemplo**] - La siguiente posición (figura 4.34), ejemplifica de manera simple la destrucción de la defensa. Las blancas ganan después de **1 ♖×h7+! ♔×h7 2 ♕h5#**. Una combinación sencilla que demuestra que el jugador en inferioridad tenía una estructura de peones que, al destruirla, lleva al mate.

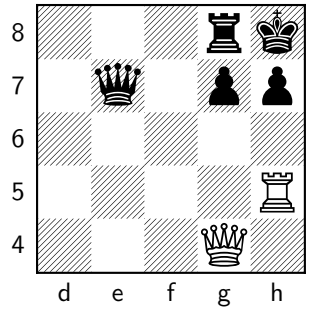

Figura 4.34: Destrucción de la estructura de peones

[**Contexto**] - La estructura de peones defectuosa es normalmente un indicio importante de las debilidades del jugador en desventaja. La destrucción de la misma es la fórmula tradicional para hacerse de la ventaja.
[**Problema**] - La detección de una estructura de peones defectuosa, sobre todo en donde se encuentra el rey es un motivo para que el jugador que busca la ventaja intente la destrucción de la misma, provocando peores debilidades o bien un ataque imposible de detener.
[**Solución**] - En general, la destrucción de la estructura de peones debe hacerse sacrificando material. La compensación por el mismo tiene que ver con un ataque poderoso contra el rey enemigo, el cual en ocasiones, si no tiene éxito, al menos garantiza el empate. Normalmente la defensa es penosa.
[**Estructura**] -
[**Dinámica**] - Normalmente la destrucción de la estructura de peones del contrario se lleva a cabo con varias piezas que coordinan el ataque. Por ejemplo, en el siguiente diagrama (figura 4.35), a pesar de que las negras parecen no tener dificultades, su rey sucumbe por la falta de posibles defensores para el mismo cuando las blancas destruyan su guarida (*aka* su estructura de peones. Nótese como las blancas sacrifican una pieza para entra en las líneas enemigas.

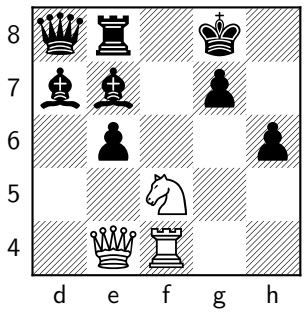

Figura 4.35: La secuencia ganadora empieza con **1 ♘×h6+ g×h6 2 ♕g6+ ♔h8 3 ♖f7** y mate a la siguiente jugada.

[**Ejemplo resuelto**] - En **Kramnik** – **Fressinet**, París (Alekhine Memorial) 2013 (ver figura 4.36)[20]. Juegan las negras, éstas triunfan destruyendo los restos de la estructura de peones. Fressinet lo demuestra con solvencia: **25. . . ♗×f2+ 26 ♔×f2 ♗×f1 27 ♔×f1 g3 28 ♗f3 g×h2 29 ♔e2 ♖hg8 30 ♗c5 a6 31 ♗h1 ♖g2+ 32 ♗×g2** 0-1.

[20]http://www.chessgames.com/perl/chessgame?gid=1715950

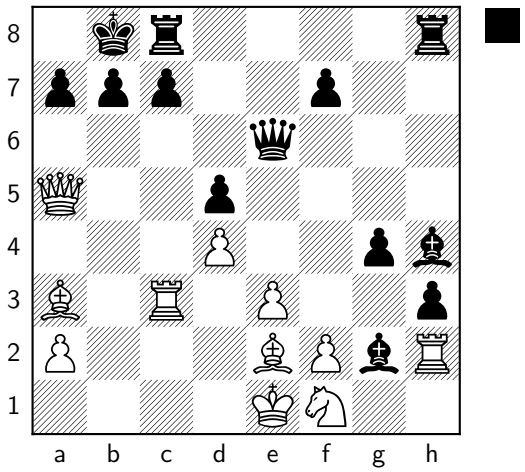

Figura 4.36: **Kramnik – Fressinet**, París 2013. Juegan las negras y ejecutan la destrucción de la estructura de peones que a la postre lleva a la victoria.

[**Variantes**] - Hay muchas variantes de este tema, por ejemplo, destruir la estructura de peones rivales simplemente avanzando los propios peones. En otras ocasiones un cambio de piezas hace que la estructura deje debilidades que son después imposibles de defender. Hay muchos ejemplos de este tipo de temas e incluso, muchas veces, no necesariamente lleva a un ataque mortal. Frecuentemente simplemente se logra una cómoda ventaja a largo plazo.
[**Usos conocidos**] - Aunque es un patrón genérico, hay muchas partidas de ataque en donde la destrucción de los peones logran minar la defensa y generar un ataque ganador. Como por ejemplo, Alekhine – Marshall, Baden Baden 1925, o Mueller, H. – Alekhine, Kecskemet (1927).
[**Consecuencias**] - La aplicación de este patrón ilustra cómo deben tratarse las posiciones en donde existen debilidades estructurales en la estructura de peones. La consecuencia de la destrucción de la posición en donde se encuentran los peones lleva en general a un ataque de consecuencias ganadoras. En términos reales, la defensa suele ser dura, difícil, y casi nunca exitosa.

4.7. Resumen

En este capítulo se analizan los patrones que son técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos. Un patrón resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. En el caso de los juegos de suma–cero, como el ajedrez, el gato o el NIM, se pueden describir una serie de patrones que bien pueden ser usados en ciertas posiciones típicas del ajedrez, gato o NIM. Así pues, hay patrones para "la fórmula de Tarrasch", la eliminación de un defensor, el regalo griego, así como para los temas clásicos como la clavada, la distracción, el rey ahogado, la destrucción de la estructura de peones y la jugada intermedia, para el caso del ajedrez, o bien los mejores primeros movimientos en el caso del gato.

Capítulo 5

Funciones de evaluación *versus* patrones

Normalmente los juegos de suma–cero e información perfecta se resuelven a través de algoritmos como Minimax o Alpha-Beta, considerando una función de evaluación y una estructura arbórea, donde se van desarrollando los nodos de las diferentes jugadas y respuestas que pueden darse a partir de una posición dada en el juego. Este enfoque no contempla el uso de patrones conocidos como ventajosos, incluso en etapas tempranas del juego, como en el juego del Gato o en el NIM. En este capítulo se analizan ambos enfoques sus pros y contras.

5.1. Función de evaluación

Una función de evaluación, conocida también como una función heurística o estática, se usa en muchos juegos para estimar el valor de bondad de una posición, particularmente cuando se usan algoritmos como Minimax [5]. En algunos juegos se puede aplicar una función de evaluación simple f a la posición P, $f(P)$ y saber si un movimiento puede determinar el estado final del juego (ganar, empatar, perder). En el juego de NIM, por ejemplo, esto puede determinarse escribiendo el número de fichas en cada fila en notación binaria. Estos números se ordenan en una columna (para añadirlos). Si el número de unos en cada columna es par, la posición está perdida para el jugador que tiene que mover. De otra forma gana.

La función de evaluación $f(P)$ puede diseñarse –en principio– de forma

que el sistema jugase el juego perfecto, sin errores. No pierde nunca y si el oponente hiciese un error, podría capitalizarlo. Podemos suponer $f(P) = +1$ si se tiene una posición ganada, $f(P) = 0$ si el juego es un empate y $f(P) = -1$ si la posición está perdida [75].

Cabe decir que el valor de la función de evaluación representa la probabilidad relativa de ganar dentro de un árbol de movimientos, en donde el nodo final se está evaluando. De hecho, la función solamente ve la posición actual y no toma en cuenta la historia de la posición. Esto quiere decir que si hay elementos tácticos, dinámicos, la función de evaluación no puede hacer una balance adecuado de la posición. A esto se le llama no-quiescente y no debe evaluarse la posición de forma estática, porque es muy poco confiable (por ejemplo, en ajedrez, en una posición determinada podemos valorar la captura de la dama enemiga y verla como adecuada, pero esta información estática puede cambiar si se extiende el horizonte de la búsqueda) [18]. Para resolver este problema, se requiere una extensión llamada "búsqueda quiescente", que básicamente tiene que ver más profundamente en el árbol de posibles jugadas.

No existen funciones de evaluación perfectas en un número de juegos y en muchos casos se van refinando, en la medida que se va entendiendo mejor el problema. Una función de evaluación, entonces, se define normalmente como una combinación lineal de varios términos con diferentes pesos, los cuales determinan el valor final de la posición evaluada [40].

5.2. NIM, un juego con toda una teoría matemática

NIM se juega por dos jugadores, A y B [16]. En una mesa se colocan un número de pilas de objetos de cualquier tipo, monedas, cerillos, etcétera. El número de pilas puede de hecho ser arbitrario pero se acuerda que todas las pilas deben tener un número diferente de objetos, vamos, ninguna pila debe ser en número de objetos igual a otra en un inicio. Llamemos a estos objetos cuentas.

Se juega de la siguiente manera: El jugador A selecciona de cualquiera de las pilas la cantidad de cuentas que quiera. Lo que es importante es que al menos debe elegir una cuenta. Es decir, no puede pasar. Entonces A elige un número de cuentas de una pila y los saca del juego. Ahora es el turno del

jugador B, el cual procede de la misma manera. Quien toma la última cuenta de la mesa pierde el juego[1].

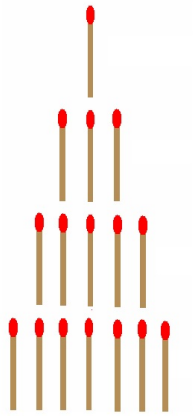

Figura 5.1: Juego de NIM de cuatro filas tradicional

5.3. NIM jugado con el algoritmo de Boulton

Supongamos el juego "estándar" de NIM, el cual puede ser de cuatro pilas, conteniendo cada una de ellas 1, 3, 5 y 7 cuentas, aunque desde luego, no hay limitaciones para la cantidad de pilas que pueden definirse.

Definamos una "combinación segura" como aquélla que se forma de escribir el número de cada pila en formato binario y colóquese en cuatro líneas horizontales de manera que coincidan las posiciones binarias, como en el siguiente ejemplo:

- 001 (1)
- 011 (3)
- 101 (5)
- 111 (7)

[1]También puede definirse quien tome la última cuenta de la mesa gana el juego.

Si la paridad de cada columna es cero, es decir, hay un número par de '1's en la columna, entonces decimos que ésa es una combinación segura. Por ejemplo, [1, 3, 5, 7] es una combinación segura (es el inicio del juego). Si ambos jugadores no cometen errores, quien empieza pierde si el juego inicia con cuatro pilas de 1, 3, 5 y 7 cuentas [15].

Las configuraciones ganadoras del NIM se resuelven a partir de lo que se llama *Suma NIM*, descrita por C. L. Boulton [15] de la Universidad de Harvard, en 1902, aunque se piensa que el juego es mucho más antiguo. La estrategia ganadora es tomar tantas cerillas como se puedan mientras la Suma NIM dé cero. Este procedimiento debe repetirse hasta el final del juego.

La Suma NIM se desarrolla de la siguiente manera: Se cuentan las cerillas en cada fila y se convierte el número en una suma de potencias de 2 (8, 4, 2 y 1). Entonces se cancelan los pares de potencias iguales y se añade el resto. Por ejemplo, el patrón inicial es 1–3–5–7:

```
Fila 1 = 1 = 1 * 1                    =     1
Fila 2 = 3 = 1 * 2 + 1 * 1            =   2 1
Fila 3 = 5 = 1 * 4 + 1 * 1            = 4   1
Fila 4 = 7 = 1 * 4 + 1 * 2 + 1 * 1 = 4 2 1
Total de potencias no par             = 0 0 0
```

Figura 5.2: Suma NIM en la posición inicial

Para ganar en el juego, hay que hacer siempre una jugada que deje la configuración de Suma NIM. Si por ejemplo, el primer jugador quita 2 cerillas de la fila 4, entonces tendremos

```
Fila 1 = 1 = 1 * 1                    =     1
Fila 2 = 3 = 1 * 2 + 1 * 1            =   2 1
Fila 3 = 5 = 1 * 4 + 1 * 1            = 4   1
Fila 4 = 5 = 1 * 4 + 1 * 1            = 4   1
Total de potencias no par             = 0 2 0
```

Figura 5.3: Siguiente posición del ejemplo

Aquí la *Suma NIM* es de 2. Los 4 se cancelan así como los 1. Por ende, el movimiento ganador debe ser quitar de la fila 2 el 2 para que así obtengamos de nuevo la Suma NIM (que debe dar cero) y de nuevo, tenemos una posición ganadora. El patrón que queda es 1–1–5–5, que es también ganador.

Mediante este procedimiento, podemos hallar todas las posibles combinaciones de patrones que son ganadores, que son Suma NIM cero y aplicarlos convenientemente en su momento para siempre hacer el movimiento ganador.

Dos teoremas formalizan esta situación [16]:

- [**Teorema 1**] Si A deja una combinación segura en la mesa, B no puede dejar una combinación segura en su siguiente jugada.
- [**Teorema 2**] Si A deja una combinación segura en la mesa y B disminuye una de las pilas, A puede entonces disminuir alguna de las pilas para dejar de nuevo una combinación segura. Jugar de esta manera al NIM requiere de cierta manipulación simbólica de las pilas y para una computadora, programar este juego es algo elemental.

Pos. inicial	B quita 1 de la 4a pila	A quita 1 de la 3a pila	B quita 1 de la 3 pila	A quita 3 de la 2a pila	B quita 1 de la 1a pila	A quita 1 de la 4a pila
001 (1)	001 (1)	001 (1)	001 (1)	001 (1)	000 (0)	000 (0)
011 (3) →	011 (3) →	011 (3) →	011 (3) →	000 (0) →	000 (0) →	000 (0)
101 (5)	101 (5)	100 (4)	100 (4)	100 (4)	100 (4)	100 (4)
111 (7)	110 (6)	110 (6)	101 (5)	101 (5)	101 (5)	100 (4)
000	001	000	011	000	001	000
paridad 0	paridad 1	paridad 0	paridad 1	paridad 0	paridad 1	paridad 0

Figura 5.4: Jugando NIM mediante la paridad binaria

Puede verse que el jugador A siempre lleva los valores a que tengan paridad cero. Este es el algoritmo general para jugar NIM con cualquier número de columnas. De hecho, debido al algoritmo conocido, en esta particular modalidad del juego, quien empieza está condenado a perder (si es que el segundo jugador siempre hace la suma NIM correcta). He aquí la gráfica que representa este resultado:

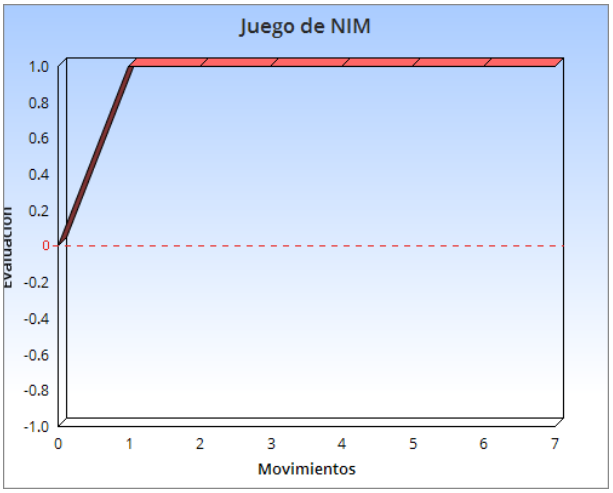

Figura 5.5: Evaluación del juego de NIM mediante el algoritmo de Boulton

5.4. NIM jugado con patrones

Jugar al NIM entre seres humanos bien puede tener otra representación (siempre y cuando se considere el juego estándar), la cual es mucho más simple de analizar que tener que pasar los valores de las pilas al formato binario y hallar la paridad. Por ejemplo, podemos partir de una representación de las pilas de la siguiente manera:

posición inicial → [1,3,5,7]

A partir de ahí, podemos identificar todas las combinaciones seguras -que podrían definirse como patrones ganadores, las cuales pueden expresarse de la siguiente manera:

1-3-5-7 →1-3-5→ **1-3-2**
1-3-5-7 →1-3-7→ **1-3-2**
1-3-5-7 →1-3-5-5→ **1-1-5-5**
1-3-5-7 →1-3-3-7→ **1-3-3-1**
1-3-5-7 →3-5-7→ **3-5-6**
1-3-5-7 →1-2-5-7→ **1-2-4-7**
1-3-5-7 →3-5-7→ **2-5-7**
1-3-5-7 →1-3-5-6→ **1-2-5-6**

1-3-5-7 →1-3-5-6→ **1-3-4-6**
1-3-5-7 →1-3-4-7→ **1-3-4-6**
1-3-4-6 →3-4-6→ **2-4-6**
1-3-4-6 →1-3-4-5→ **1-4-5**
1-3-4-6 →1-2-4-6→ **2-4-6**
1-3-4-6 →1-2-3-6→ **1-2-3**
1-3-4-6 →1-3-2-6→ **1-3-2**
1-3-4-6 →1-3-4→ **1-3-2**
1-3-4-6 →1-1-4-6→ **1-1-4-4**
1-3-3-1 →1-3-3→ **1-3-2**
1-3-3-1 →1-3-3→ **3-3**
1-2-2-1 →1-2-2→ **2-2**
1-1-5-5 →1-5-5→ **5-5**
1-1-5-5 →1-1-5→ **1-1-1**
1-1-5-5 →1-1-4-5→ **1-1-4-4**
1-1-4-4 →1-1-4→ **1-1-1**
1-1-4-4 →1-1-3-4→ **1-1-3-3**
1-1-4-4 →1-1-2-4→ **1-1-2-2**
1-1-3-3 →1-1-3→ **1-1-1**
1-1-3-3 →1-3-3→ **1-3-2**
1-1-2-2 →1-1-2→ **1-1-1**
1-1-2-2 →1-2-2→ **2-2**
3-5-6 →2-5-6→ **2-4-6**
3-5-6 →1-5-6→ **1-5-4**
3-5-6 →3-4-6→ **2-4-6**
3-5-6 →3-5-5→ **5-5**
3-5-6 →3-5-4→ **1-5-4**
2-4-6 →2-4-5→ **1-4-5**
2-4-6 →1-4-6→ **1-4-5**
2-4-6 →2-4-4→ **4-4**
2-4-6 →4-6→ **4-4**
2-4-6 →2-6→ **2-2**
1-2-3 →1-2-2→ **2-2**
1-2-3 →1-2-1→ **1-1-1**
1-2-3 →1-1-3→ **1-1-1**
1-4-5 →1-4-4→ **4-4**
1-4-5 →4-5→ **4-4**
1-4-5 →1-3-5→ **1-3-2**

1-2-3 →1-2→ **1-0**
1-1-1 →1-1→ **1-0**
5-5 →5-4→ **4-4**
5-5 →5-3→ **3-3**
5-5 →5-2→ **2-2**
5-5 →5-1→ **0-1**
4-4 →4-3→ **3-3**
4-4 →4-2→ **2-2**
4-4 →4-1→ **0-1**
3-3 →3-2→ **2-2**
3-3 →3-1→ **0-1**
2-2 →2-1→ **0-1**
2-2 →2-0→ **1-0**

Cada combinación segura puede llevarse a una que tiene eventualmente menos número de pilas. En el caso de la figura 5.4, puede hallarse la siguiente secuencia de patrones: [1,3,5,7] → [1,3,5,6] → [1,3,4,6] → [1,3,4,5] → [1,4,5] → [4,5] → [4,4] → [3,4] → [3,3], etcétera.

El autor del artículo [15] resuelve el caso general del juego de NIM a partir de utilizar la idea de pasar a binario los valores de las pilas y analizar la paridad de éstas para saber si se tiene una combinación segura. Este enfoque suena sencillo de aplicar cuando se trata de escribir un programa de computadora que resuelva el juego, pero no trata el problema utilizando configuraciones ganadoras (patrones), los cuales son fáciles de recordar por jugadores humanos.

Sin embargo, es posible analizar cualquier tipo o variante de juego del NIM, más o menos pilas, más o menos cuentas por pilas, y hallar entonces las combinaciones seguras. En ese sentido la resolución de Charles Boulton [15] resulta completa y general.

5.5. La evaluación en el juego del gato

Uno de los juegos más simples de estudiar es el gato (*tic-tac-toe*) –ver sección 4.1, sin embargo, ilustra muy bien el tema de la evaluación de las configuraciones que se producen. Es evidente que la función de evaluación es única para cada juego, sin embargo, en cualquier caso la idea detrás de esta función es maximizar la valoración en el turno de un jugador mientras

se minimiza el del contrario [1]. En el algoritmo Minimax se toma la función de evaluación de los nodos hasta que se llega al nodo terminal, y de ahí se regresa recursivamente al inicio, haciendo Max y Min en cada nivel, para decidir –a partir del nodo terminal– la valoración de cada jugada.

Para el caso del juego del Gato, tomaremos la función $f(P)$, que puede tomar 3 posibles valores: $+1$, 0 y -1, ganando, empatando o perdiendo para el primer jugador, respectivamente. Usaremos el código ejemplo en C++[2].

Si analizamos el caso de que ambos jugadores hacen sus mejores movimientos, observaremos lo siguiente:

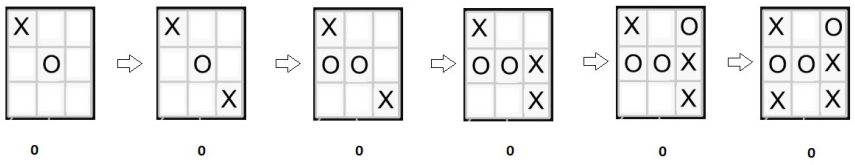

Figura 5.6: Juego perfecto de Gato, por ambos jugadores

[2]https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/?ref=rp

La gráfica muestra la igualdad de la valoración en cada movimiento (figura 5.7).

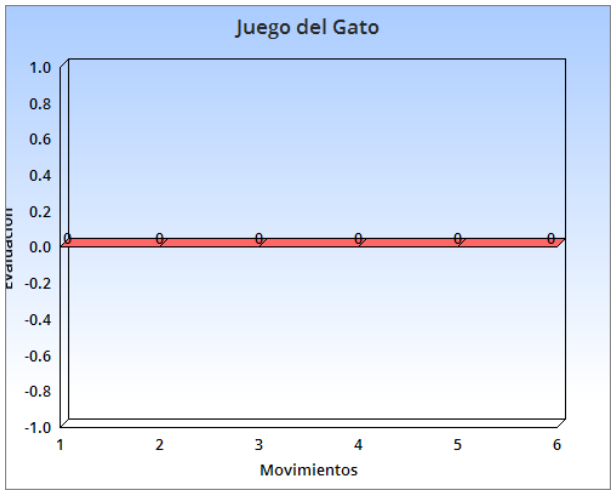

Figura 5.7: Juego perfecto de Gato, por ambos jugadores

Cabe destacar que la elección de la mejor jugada solamente se conoce cuando se llega a los nodos terminales, que son los que definen el resultado final de cada jugada, después de la generación del árbol de posibilidades, el espacio de búsqueda.

Para el caso de un juego defectuoso (o erróneo) por el segundo jugador, se tiene la siguiente valoración (figura 5.8):

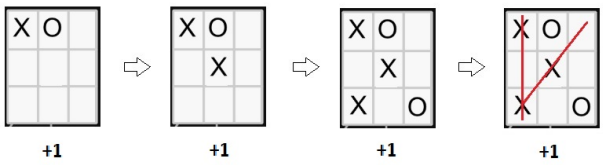

Figura 5.8: Juego perfecto de Gato, por ambos jugadores

En este caso, el primer jugador gana. Hay que decir que se espera el mejor movimiento de cada jugador. En caso de un error de alguno de ellos, el otro puede capitalizar su ventaja (figura 5.9).

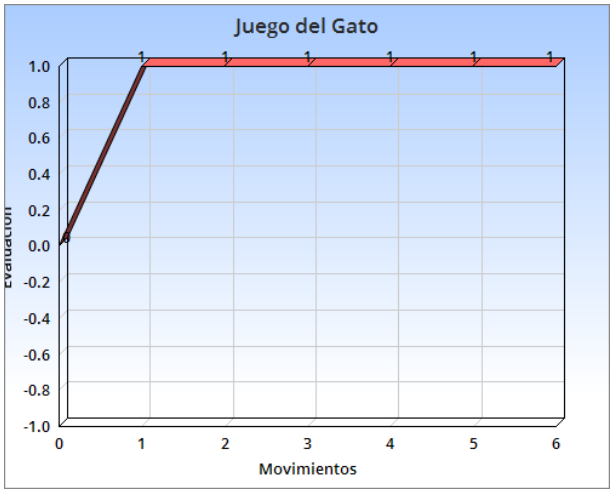

Figura 5.9: Capitalización de la ventaja por parte del primer jugador

5.6. Usando patrones para resolver el juego del gato

A pesar de su simpleza, pueden definirse una serie de patrones ganadores elementales, los cuales incluso cualquier niño aprende al poco tiempo de que se le enseña el juego. Asumiendo que empieza la X inicialmente, tenemos tres posibles alternativas (ver figura **??**).

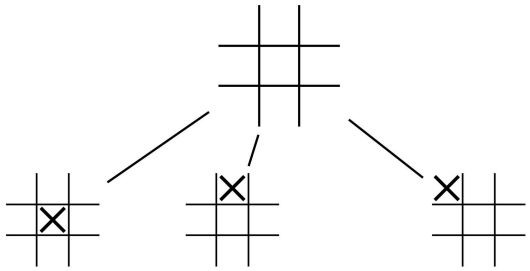

Figura 5.10: Los tres primeros patrones (chunks), del juego del gato en el primer movimiento

A partir de estas tres posibles configuraciones, el rival, la O puede contestar de diversas maneras (ver figura 5.11).

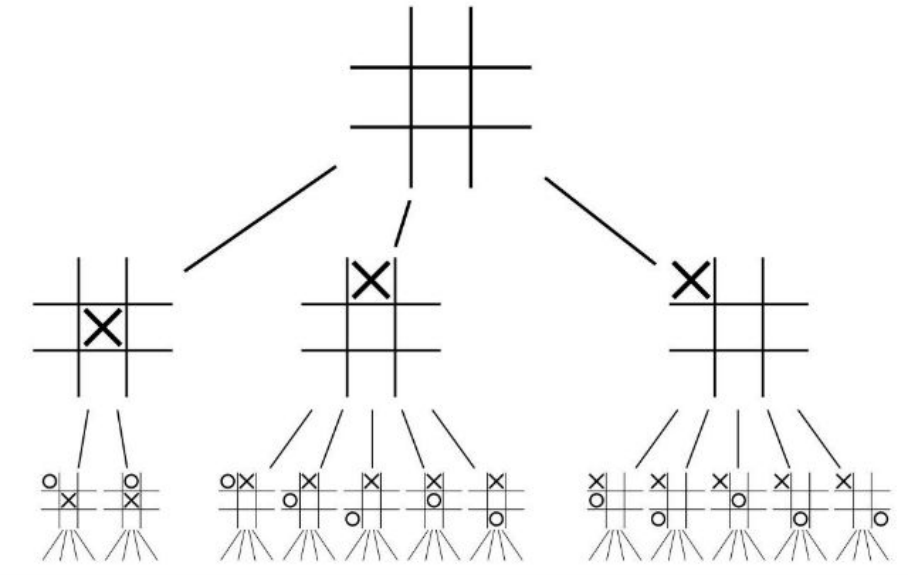

Figura 5.11: Las posibles respuestas del rival a los tres patrones iniciales

Podemos decir que no hay que mostrar todas las alternativas de "O", sino que se saca provecho de que el juego es simétrico, es decir, tirar en cualquiera de las esquinas tiene un resultado equivalente. Además, hay que señalar que del total de posibles combinaciones 9!, muchas son inválidas, es decir, no se puede llegar a ellas porque cuando uno de los jugadores hace tres en línea, el juego termina.

La cantidad de posibles posiciones legales del juego del gato, incluyendo el tablero vacío, es de 5478 configuraciones, tomando en cuenta aquéllas que son simétricas[3]. El juego se puede analizar examinando todas las ramas del árbol de posibles variantes dentro del juego, es decir, de las configuraciones de "X" y "O" que pueden aparece. Si vemos los nodos terminales de cada juego podemos saber quién gana, empata o pierde.

En principio, a partir de la siguiente figura (5.12), se puede encontrar quien gana.

[3]http://www.se16.info/hgb/tictactoe.htm

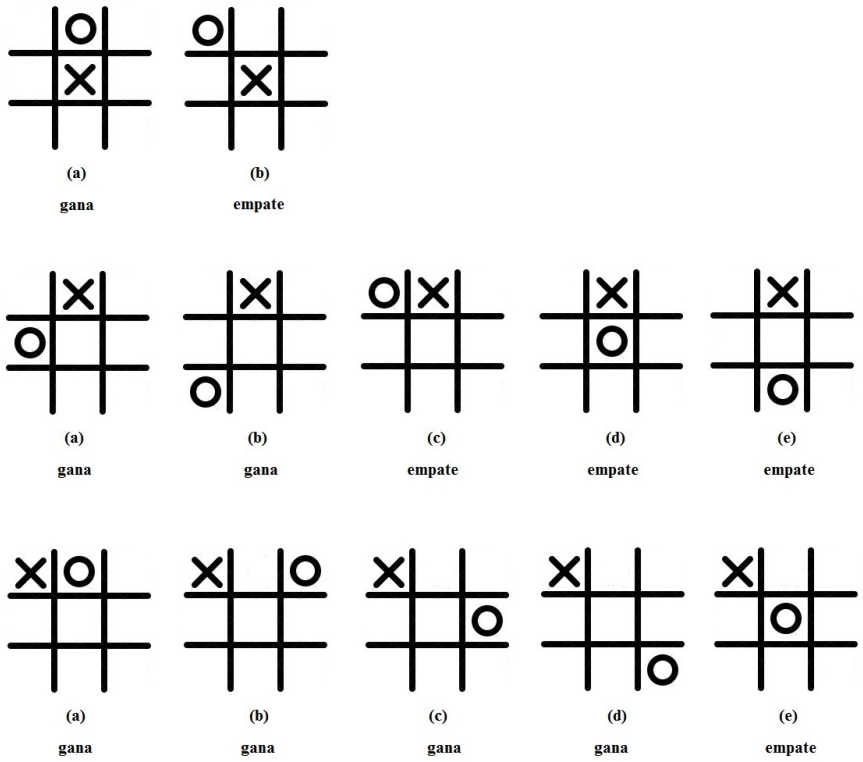

Figura 5.12: Todos los patrones del gato, ganadores y de empate, asumiendo que empiezan las "X".

Es posible escribir un programa que juegue todos los posibles juegos del gato. Si eliminamos las configuraciones repetidas (es decir, que se llega a una posición determinada en el tablero desde dos diferentes conjuntos de movimientos) y las simétricas, entonces nos quedan 765 posiciones[4] "legales" del gato, definiendo éstas como posiciones que se pueden dar en un juego que siga las reglas del mismo.

Hay que decir que las configuraciones o patrones ganadores en el juego del gato, aunque estos se den desde la segunda jugada, son **instantáneos**, es decir, la ventaja ganadora se da en ese momento y el jugador que tiene esa posición y que le toca jugar, debe hacer los movimientos correctos para ganar. Dicho de otra manera, el camino al triunfo solamente existe si se mantiene

[4]Un programa escrito en Haskell que resuelve esta tarea puede verse en http://brianshourd.com/posts/2012-11-06-tilt-number-of-tic-tac-toe-boards.html

la ventaja haciendo los movimientos correctos. Si no se da esto, la ventaja puede desaparecer, también de forma instantánea. Sin embargo, se presupone que siempre ambos jugadores hacen los mejores movimientos siempre.

La siguiente secuencia (figura 5.13) muestra este caso. Aquí, el conductor de las "X" hace todos los movimientos correctos después del primer movimiento de las "O", que es un error y que pierde:

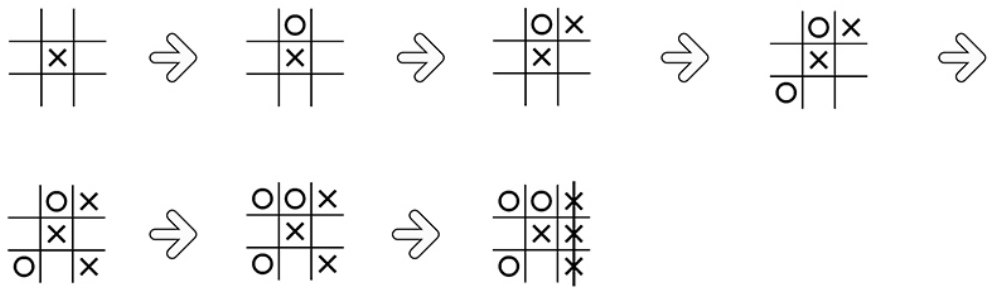

Figura 5.13: Secuencia ganadora a partir del error inicial del segundo jugador

5.6.1. Patrones del gato y su manipulación simbólica

En la figura 5.14 que identifica los casilleros del juego del gato, podemos representar entonces cualquier posición.

Podemos expresar cada posición como una lista, por ejemplo, la posición siguiente (ver figura 5.15) es ganadora para el jugador que lleva las "X": [x, o, b, b, b, b, b, b, b], donde la "b" representa una casilla vacía. A esto le llamamos un *estado* o también un *patrón*.

Además de esto, tenemos una manera de representar los movimientos de los jugadores, en estricta secuencia de izquierda a derecha: *[X—O]número*. Por ejemplo, para la figura 5.15, los dos primeros movimientos pueden expresarse como X1 $\rightarrow$ O2.

Definimos una *transformación* cuando hay una secuencia de movimientos que llevan de un *estado inicial ventajoso* a un *estado ganador*:

$T : E_i \rightarrow E_f$

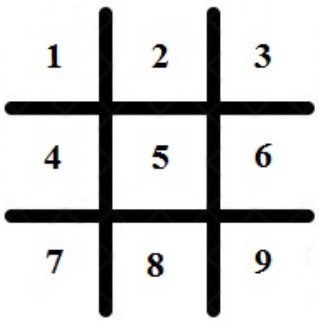

Figura 5.14: Las coordenadas en el juego del gato

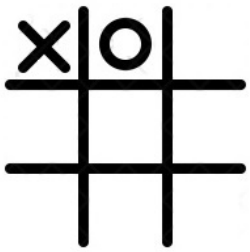

Figura 5.15: Una de las configuraciones ganadoras más simples

donde E_i es el estado inicial y E_f es el estado final al cual se quiere llegar. Esta transformación pude verse como una secuencias forzada de estados $s_1, s_2, ..., s_n$.

A partir de esto se puede decir que del patrón ventajoso (véase figura 5.15), hay una transformación T definida como X1 $\rightarrow$ O2 $\rightarrow$ X5 $\rightarrow$ O8 $\rightarrow$ X7 ganando en el siguiente movimiento, que lleva de un estado de ventaja a un estado ganador.

Así, P = [x, o, b, b, b, b, b, b, b] y T = X1 $\rightarrow$ O2 $\rightarrow$ X5 $\rightarrow$ O8 $\rightarrow$ X7 que conduce a un patrón ganador.

Nótese –de nuevo– que el jugador de las "X" hace todos los movimientos correctos y por ende, el rival no puede salvar la partida después de su primer movimiento, el cual pierde (si se juega correctamente).

5.7. Funciones de evaluación en ajedrez

Fue Shannon quien define formalmente una función de evaluación muy sencilla, en donde se ve el balance del material el cual domina, muchas veces, la evaluación, además de una serie de términos posicionales, cada uno de ellos no excediendo el valor de la unidad del juego, definida como el peón [75].

He aquí la función de evaluación definida por Shannon:

$f(P) = 200(R - R') + 9(D - D') + 5(T - T') + 3(A - A' + C - C') + (P - P') - 0{,}5(D - D' + S - S' + I - I') + 0{,}1(M - M') + ...$
en donde

- R,D,T,A,C,P es el número de reyes, damas, alfiles, torres, caballos y peones en el tablero.
- D,S,I son peones Doblados (D), retrasados (S) y aislados (I).
- M = se define como la movilidad, medida como el número de movimientos posibles para las blancas[5].

Los coeficientes 0.5 y 0.1 son estimaciones empíricas de Shannon y hoy en día podrían no ser aplicables, al menos en toda la partida. La fórmula pretende ilustrar la función y para los estándares actuales puede ser demasiado simple. Cabe señalar que el valor del rey es siempre mucho mayor al de otras piezas (200 puntos en el ejemplo), pues sin éste el juego no existe, por lo cual pudiese quitarse [75].

5.8. Evaluación de las posiciones en ajedrez

El ajedrez no puede ser resuelto totalmente, es decir, saber el resultado final en cada posición. El crecimiento de posibilidades es alrededor de 10^{43} [75]. Por ello se usa una función heurística que determina el valor relativo de las posición, es decir, las chances de ganar –suponiendo que pudiésemos ver el final de la partida. Ya hemos hablado que esto sería definir los valores de $+1$, 0 y -1, respectivamente ganar, empatar y perder. Pero como esto nos es posible, lo que las funciones de evaluación hacen es dar una aproximación.

[5]Las letras primadas son las cantidades similares pero para las negras

El resultado de esta evaluación depende, desde luego, de los términos que contenga la función lineal[6].

5.9. Valoración del "regalo griego"

A través de la función de evaluación, los programas de ajedrez modernos valoran las diferentes posiciones y dan un estimado numérico como resultado de esto. La unidad –como ya se dijo– es el peón, y la función de evaluación mide hasta centésimas de peón. De esta manera, +0,54 significaría, por ejemplo, que las blancas tienen una ventaja equivalente a medio peón. En cambio, −2,93 querría decir que las negras tienen una ventaja equivalente a casi tres peones (lo cual en términos generales significaría que tiene una posición ganadora). Usando dos programas modernos y muy fuertes: Stockfish 11 y Fire 7.1, se analiza el llamado regalo griego y veremos la evolución de la gráfica que muestra como uno de los jugadores obtiene la ventaja.

El ejemplo corresponde a la partida entre

Bianchi, Guillermo (2430) – Haitshandiet, Miguel
Buenos Aires GP II (2) 1992

1 e4 e6 2 ♘f3 d5 3 ♘c3 ♘f6 4 e5 ♘fd7 5 d4 ♗e7 6 ♗d3 a6 7 ♘e2 c5 8 d×c5 ♘c6 9 O-O O-O 10 ♖e1 ♖e8 11 ♘f4 ♗f8 12 h3 ♘×c5 13 ♗×h7+ ♔h8 14 ♘g5 ♘×e5 15 ♕h5 g6 16 ♗×g6+ ♔g7 17 ♘g×e6+ ♖×e6 18 ♕h7+ 1-0

Usando *Stockfish*[7], la gráfica correspondiente a esta partida muestra claramente el encuentro está relativamente igualado hasta la jugada 15, donde las blancas obtienen una ventaja de casi 3 puntos. Sin embargo, cometen un error definitivo y la valoración de la posición se va a casi +5 en favor de las blancas, que terminan ganando. El sacrificio de la pieza, el *regalo griego*, resulta ganador aunque las negras juegan muy deficientemente esta parte de la partida.

[6]Un documento de 23 páginas, actualizado al 06.04.2010, muestra los elementos más significativos de las funciones de evaluación usadas por los diferentes programas a lo largo de la historia del ajedrez por computadora: Véase http://www.winboardengines.de/doc/LittleChessEvaluationCompendium-2010-04-07.pdf

[7]Un motor de análisis de ajedrez de código abierto y de los mejores hoy día.

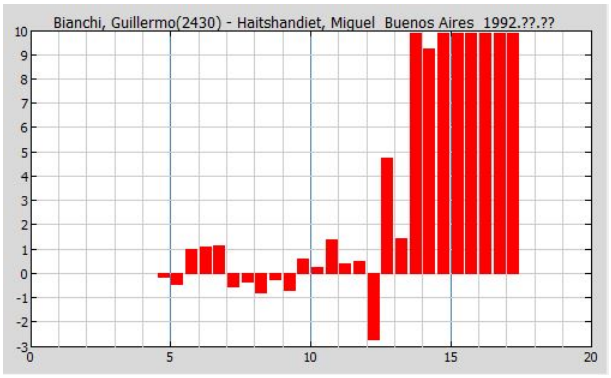

Figura 5.16: **La valoración de Stockfish en el regalo griego, en la partida Bianchi – Haitshandiet**

Mientras que *Fire 7.1*[8] valora de la siguiente manera:

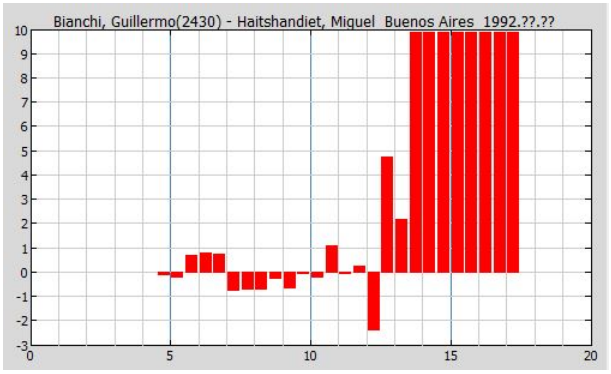

Figura 5.17: **La valoración de Fire 7.1 en el regalo griego en la partida Bianchi – Haitshandiet**

Puede verse que Stockfish y Fire 7.1 valoran las posiciones de manera parecida pero no idéntica. La razón de ello es que la función de evaluación de cada programa es diferente, aunque de manera sutil. En ambas gráficas puede verse que las negras cometen un error en la jugada 15 y pierden no solamente su ventaja, sino que quedan perdidas totalmente.

[8]Este es uno de los motores de análisis de posiciones de ajedrez popularmente usado por los aficionados al ajedrez.

5.10. Patrones en las posiciones

En el juego de ajedrez hay situaciones estratégicas similares, en donde el número de piezas varía, el conjunto de las mismas varía, el tipo de piezas varía, la posición de las piezas varía y el árbol de búsquedas varía en profundidad y ramaje [4]. Es decir, son posiciones diferentes pero similares. La literatura ajedrecística está llena de ejercicios de táctica, extraídos en general de partidas de torneo, donde se repiten las mismas características, lo cual llamamos en este contexto "patrones". Llama la atención algunos ejemplos notables que son comunes en las partidas de los grandes maestros. Por ejemplo, el gran maestro Rosentalis comenta en una de sus partidas: "Cuando jugué ♕**a3** llegó a mi mente la partida Smyslov – Reshevsky" [79]. Sin embargo si se revisan ambas partidas, las dos posiciones de interés no tienen ningún parecido visual obvio y aún así Rosentalis percibe ambas posiciones compartiendo algunos aspectos cruciales que le permite, basándose en estas similitudes, hallar la jugada ♕**a3** (permitiendo el cambio de damas).

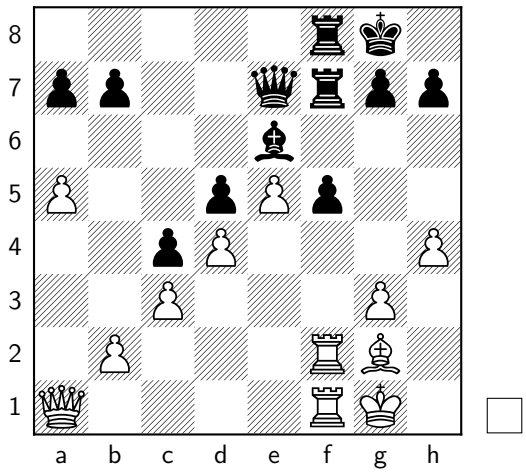

Figura 5.18: **Rosentalis** – **Appel**, Bundesliga 1993

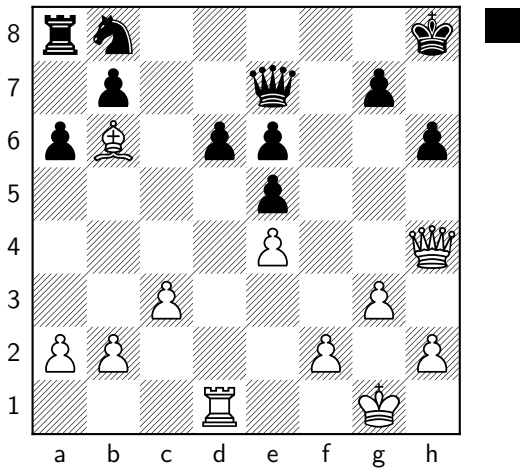

Figura 5.19: **Smyslov** − **Reshevsky**, Campeonato Mundial 1948

Sabemos, por los estudios de Adriaan de Groot [29], que la concepción básica de lo que es tener un nivel de experto en ajedrez se reduce a hallar las mejores jugadas basándose en reconocimiento y asociación [48], y hay suficiente cantidad de datos para indicar que los jugadores experimentados tienen acceso a una gran base de datos de patrones, también llamados *chunks*, y estos están asociados con planes e ideas plausibles [49]. Un chunk en ajedrez se define como una unidad de información que contiene un grupo de piezas agrupados con algún sentido, el cual además se asocia a movimientos e ideas [79]. Y mas aún, cada chunk consiste en hasta cinco piezas y el tamaño del chunk guardado está relacionado positivamente con la destreza en el juego.

En el ejemplo anecdótico de Rosentalis se muestra un dato importante sobre la teoría de los chunks, que es que no necesariamente se basan en la información de las piezas en sus posiciones. Linhares propone que la investigación correspondiente para el ajedrez no se debe enfocar en la parte visual del juego [58][9], y aunque en muchas situaciones hay similitudes en las posiciones (chunks) es evidente que parece haber un nivel de abstracción de dichos patrones que están relacionados directamente con el nivel de maestría en el juego del ajedrez.

En un experimento diseñado por Linhares y Brum [4], se construyeron posiciones que podían agruparse de forma visual o basándose en principios

[9]Disponible en http://cogprints.org/6615/

abstractos. Más allá de los detalles de dicho experimento, en donde ponen a jugadores de diversos niveles, desde principiantes a maestros de ajedrez a agrupar posiciones, se encuentra que los jugadores expertos agruparon las posiciones en pares abstractos, mientras que los aficionados y principiantes solamente aparejaron la mitad de las posiciones, es decir, no notan la similitud en la abstracción. Más aún, prácticamente ningún experto hace algún agrupamiento basado en alguna concepción de similitud visual. Los investigadores concluyen que hay diferentes formas de codificar posiciones de ajedrez, desde representaciones superficiales hasta aquellas que usan una semántica abstracta. De acuerdo con estos investigadores, lo que diferencia a un experto de un novato es el nivel de abstracción con la que representan a las posiciones.

La representación visual de los chunks es importante para definir los patrones de posiciones ventajosas, por ejemplo, pero una representación mucho más amplia puede darse utilizando el lenguaje de descripción de patrones de ajedrez que se ha definido en el capítulo 5. Una ventaja del mismo es que le quita a los chunks, a las configuraciones ganadoras, a los patrones, la necesidad de la información visual, manteniendo las características que fundamentan el chunk en cuestión.

Bibalić y Gobet [47] critican el trabajo de Linhares y Brum, indicando que los expertos no agrupan las parejas de posiciones porque pensaran que era la forma natural de hacerlo, sino porque se les instruye precisamente a hacerlo de esa manera. Linhares y Brum, sin embargo, responden a esta crítica indicando que los expertos bien podrían comportarse como novatos pero *los novatos no pueden comportarse como los expertos.* [57]

Pero más allá de estas críticas, es claro que hay una abstracción que los expertos pueden hacer, basándose en las características de las configuraciones de piezas presentadas. Parece ser además claro que esta representación es más abstracta en la medida que el nivel del ajedrecista es mayor [57].

5.11. Resumen

En este capítulo se analizan algunos juegos de suma–cero e información perfecta a través del uso de patrones. Puede verse que los resultados obtenidos muestran que el uso de patrones ganadores o ventajosos es suficiente para guiar las jugadas de los adversarios. Se revisan el juego del gato, el de NIM y el ajedrez, sacando las conclusiones de cada caso.

Capítulo 6

Análisis de algunos de los patrones ventajosos más significativos en ajedrez

El ajedrez contiene una serie de patrones ventajosos muy conocidos. En el presente capítulo se analizan algunos de los mismos más representativos. Hay un número de juegos que cumplen con esta definición, algunos ya resueltos, que pueden ayudar a ilustrar el uso de los patrones ganadores.

6.1. El juego del ajedrez

El núcleo de este trabajo es demostrar que en los juegos de suma–cero hay patrones que pueden identificarse como ventajosos (o bien ganadores), incluso en etapas tempranas del juego. El caso de estudio se centra en el ajedrez, uno de los pocos juegos de suma–cero que no está resuelto definitivamente pero que ha sido objeto de estudio por muchísimos años. El ajedrez tiene la virtud de que ya existe una colección de partidas que pueden ser leídas por los programas de computadora en un formato que es universal para este juego[1].

El ajedrez es un juego de estrategia en el que el objetivo es acabar, "ma-

[1]Se utiliza el formato PGN – *Portable Game Notation*, el cual es un formato de texto sin caracteres especiales, muy fácil de utilizar. Hoy en día las bases de partidas –en formato electrónico– rebasa los ocho millones, aglutinando toda la historia del juego–ciencia a partir de los primeros registros, que datan del año 1560, aproximadamente.

tar", al rey del oponente. Esto se hace amenazando la casilla que ocupa el rey con alguna de las piezas propias sin que el otro jugador pueda proteger a su rey interponiendo una pieza entre su rey y la pieza que lo amenaza, mover su rey a una casilla libre o capturar a la pieza que lo está amenazando, lo que trae como resultado lo que se denomina *jaque mate* y el fin del juego.

Éste es un juego en donde las configuraciones de piezas de ambos jugadores se repiten en muchas ocasiones. Estas posiciones comparten elementos comunes a los cuales llamamos patrones. Curiosamente estos elementos pueden ocurrir en posiciones que parecen ser diferentes, pero un análisis más profundo muestra que los elementos comunes se repiten. Y así como se repiten estas configuraciones, la manera en cómo tratarlas también se repiten, haciendo así los movimientos correctos.

La literatura de ajedrez está llena de libros con ejercicios de táctica donde los autores ya han encontrado las configuraciones, los patrones de ajedrez, en donde de una u otra forma, se repiten los mismos elementos, lo cual da entonces una ventaja al jugador que conoce el patrón porque ya sabe cómo actuar sin necesidad de hacer largos análisis en el árbol de variantes.

El ser humano puede, con estudio y práctica constantes, empezar a generar un catálogo mental de estas posiciones ganadoras (o donde se obtiene una ventaja significativa). Sin embargo, no está claro cómo el cerebro reconoce estos patrones existentes en muchas posiciones que son diferentes en apariencia pero que tienen elementos comunes que eventualmente describen uno o más patrones conocidos.

Las computadoras, por otra parte, no juegan ajedrez a través de patrones, sino que sacan ventaja de su poder y velocidad de cálculo. Esto permite hacer análisis de las posiciones basándose en una *función de evaluación* que revisa cada una de las posiciones que pueden ocurrir en una partida de ajedrez. Esto no saca pues ventaja de la naturaleza del juego de ajedrez, de los patrones que aparecen en el tablero, que son conocidos desde hace muchos años.

6.1.1. Un lenguaje para la descripción de los patrones de ajedrez

Hay muchas formas de describir las posiciones en ajedrez. Una de las más populares es llamada FEN (*Forsyth-Edwards Notation*). Propuesta en 1883, el esquema permite describir una posición en el tablero de forma precisa, sin ambigüedades.

Un lenguaje de descripción de patrones requiere más información de la posición en el tablero. El elemento fundamental es el ataque o la defensa. Los jugadores notan cuando una pieza está siendo atacada o defendida. En general, se dice cuando una pieza ataca a otra sin necesariamente decir en qué casilla se encuentra la pieza atacante y la pieza atacada. Esta es la forma en que los jugadores recuerdan las posiciones. No hablan en general de casillas específicas, sino solamente qué pieza ataca o defiende. Es decir, dan información incompleta que el ajedrecista completa por la experiencia de su visión en el tablero.

Desde Adriaan de Groot [29], se sabe que los jugadores de ajedrez no ven las posiciones como fotografías, sino que, en lugar de eso, el cerebro codifica esto como un conjunto de relaciones entre las piezas involucradas. Esta es la razón por la cual, cuando de Groot les muestra a los ajedrecistas en una prueba posiciones sin lógica entre las piezas, los resultados para reproducir las posiciones fueron igual de malos en los aficionados y los maestros. El estudio de De Groot elimina el mito de que en ajedrez se tiene una memoria fotográfica del tablero y además, confirma que de alguna manera los ajedrecistas recuerdan las posiciones a partir de las relaciones entre las piezas en el tablero.

Se puede definir un patrón en ajedrez como una generalización de un objeto (ajedrecístico), que contiene características esenciales. Hay dos tipos de relación: un **ataque**, que ocurre cuando una pieza de un jugador ataca a una del rival y una **defensa**, cuando una pieza de un jugador se relaciona con otra pieza del mismo color. Cabe señalar que la descripción de los patrones de ajedrez no pueden verse como posiciones en donde las piezas están en casillas específicas, sino que lo que interesa es qué pieza ataca/defiende a qué otra pieza.

La definición aquí planteada es estática. Un patrón tiene que describirse también de forma dinámica, y esto se hace expresando los movimientos o secuencia de estos que le pueden dar la ventaja a un jugador. Consecuentemente, el lenguaje de descripción de patrones de ajedrez requiere de ambos factores:

1. La definición de las características que permiten a los jugadores hallar los elementos comunes de manera ue puedan compararse contra otras posiciones que son aparentemente diferentes pero que exhiben los mismo elementos comunes (y es cuando hablamos de un patrón definido).

2. La secuencia de movimientos que permiten a uno de los jugadores ob-

tener la ventaja en la posición que a la postre puede ser definitiva para ganar la partida.

El lenguaje de descripción de posiciones permite generalizar los patrones diferentes hallados en las configuraciones de piezas dadas en el tablero, permitiendo una manera simple de definir los elementos comunes en las diferentes posiciones. Una vez que estos patrones están definidos, es posible generar un catálogo de ellos lo cual podría permitir un nuevo enfoque a los programas de ajedrez y a la manera en como evalúan las posiciones. Esto significa que un programa podría guiar su análisis de manera que no tuviese que revisar todo el árbol de variantes, sino buscando patrones que den la ventaja, eliminando las posibilidades que no son prometedoras en este sentido.

6.1.2. Definición del lenguaje de descripción de posiciones

El lenguaje de descripción de las posiciones en ajedrez se define de la siguiente manera: el ataque de la pieza 1 a la 2 se escribe como:

A(B)

Si hablamos de un ataque, entonces la Pieza A es de diferente color que la Pieza B. Si son del mismo color decimos entonces que se trata de una defensa entre ambas piezas. Nótese que en esta notación definida, se puede indicar, por ejemplo, **A(ph7)** en donde se habla de una alfil blanco que ataca a un peón en la casilla h7[2]. En algunas ocasiones es suficiente definir a una pieza atacando una casilla vacía. Por ejemplo **A(h7)** indica que hay un alfil atacando la casilla h7. Es posible además describir dónde se encuentra una pieza en particular. Por ejemplo **Ah7** indica que hay un alfil blanco en al casilla h7. Nótese pues que **A(ph7), A(h7) y Ah7** son tres diferentes representaciones en el lenguaje de patrones.

Es importante hacer notar que en la descripción de los patrones en ajedrez, no se ven las posiciones de las piezas en casillas específicas necesariamente, sino lo que se describe es qué pieza ataca/defiende a otra. Además de describir el patrón que contiene los elementos fundamentales de la posición

[2]Las iniciales de las piezas en mayúsculas representan a las piezas blancas y en minúsculas representan a las piezas negras.

que lleva al jugador de interés a obtener la ventaja, también hay que indicar al menos la primera jugada para lograr esto. Un primer ejemplo puede mostrar la definición de los patrones en las siguientes posiciones:

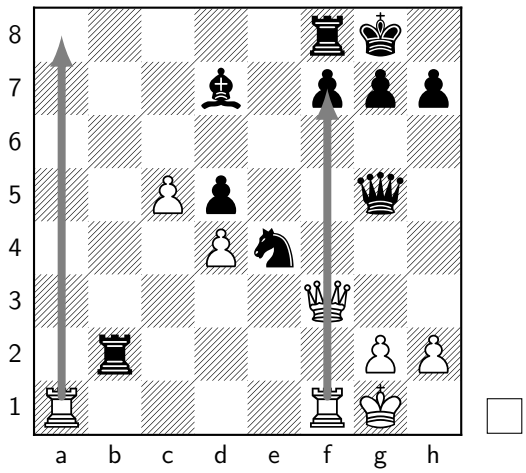

Figura 6.1: **McNab - Mullen**

Cuyo patrón es: **D(pf7), T(pf7), T(a8), 1 ♕×f7**

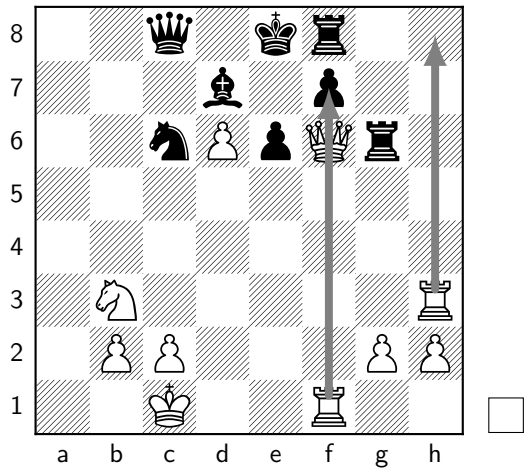

Figura 6.2: **Atakisi - Ruck**, EU-ch 4th Istanbul 2003

Cuyo patrón es: **D(pf7), T(pf7), T(h8), 1 ♕×f7**

Puede verse que en estos dos diagramas hay elementos comunes que se comparten. Por lo tanto, se puede definir esto como D(pf7), T(pf7) T(*8), donde T(*8) significa una torre que puede moverse por algunas de las 8 columnas disponibles, por ejemplo, a8, b8, c8, ... h8.

De forma general, un patrón en ajedrez descrito en este lenguaje es un conjunto de piezas atacando y defendiendo en donde se pueden encontrar elementos importantes que son los que llamamos *elementos comunes*. En algunos casos todos estos elementos pueden estar presentes pero en otros, puede ser que exista solamente un subconjunto de ellos. Esto sin embargo, define que el patrón tiene similitudes.

El siguiente ejemplo puede mostrar más estas similitudes.

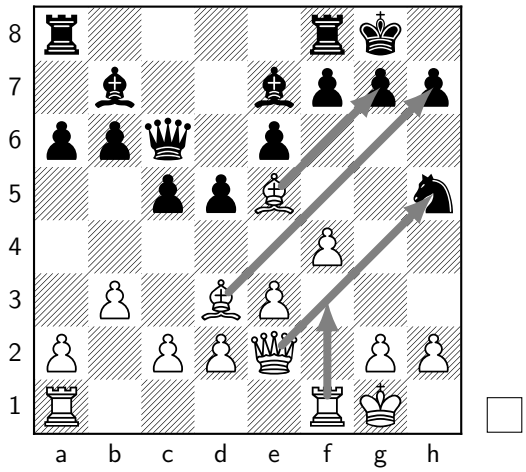

Figura 6.3: **Lasker - Bauer**, Amsterdam 1889

Patrón: **D(h5), A(ph7), A(pg7), T(f3), 1 ♗×h7+**

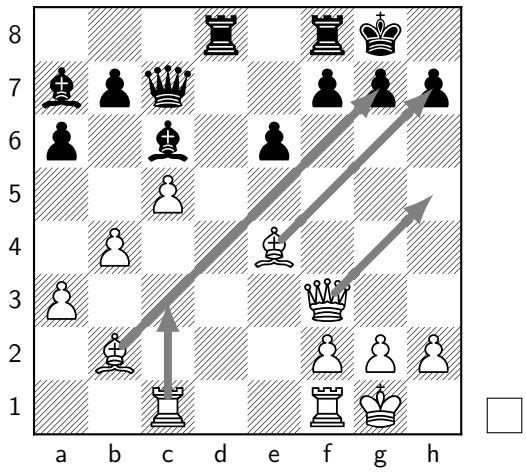

Figura 6.4: **Miles, A. - Browne, W.**, Luzern (ol) 1982

Patrón: **D(h5), A(ph7), A(pg7), T(c3), 1 ♗×h7+**

Si se observa con cuidado, se puede ver que se trata del mismo patrón en ambas posiciones. Nótese que no son estrictamente posiciones idénticas pero las similitudes son los suficientemente evidentes para cualquier jugador de ajedrez.

Puede pues decir que se trata de un lenguaje de descripción que puede (o al menos lo intenta) describir posiciones que no son necesariamente idénticas, pero sí similares. En otras palabras, hay un margen de similitud (difuso) entre ellas.

Hay autores que han propuesto una definición de patrones basándose en su similitud. De acuerdo con Peter Frey [41], se proponen tres tipos: el *Tipo-A*, donde las características de un patrón se cumplen totalmente; los del *Tipo-B*, en donde las características usualmente están presentes peor no siempre lo están y finalmente, los del *Tipo-C*, cuyas características se presentan ocasionalmente.

Hay que indicar que el lenguaje también debe poder lidiar con casillas prohibidas, es decir, inaccesibles –normalmente al jugador que se defiende. Por ejemplo, la siguiente posición es ilustra esto:

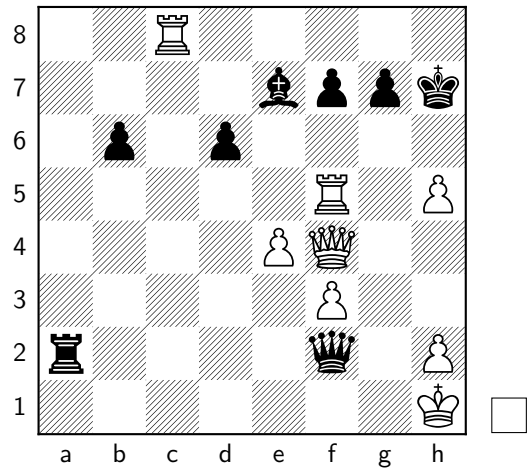

Figura 6.5: **Carlsen, M. - Karjakin, S.**, Campeonato Mundial 2016

Patrón: **D(h6), Tc8, T(f7), Ph5, rh7, pg7, taboo(g6) 1 ♕h6+**

Donde taboo(g6) indica que la casilla g6 no está disponible (desde el punto de vista de las negras).

Pero este mismo patrón ya había aparecido en la siguiente partida:

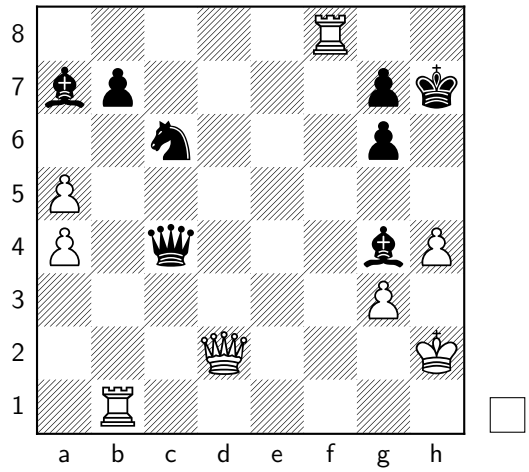

Figura 6.6: **Popov, N. - Novopashin, A**, URS (ch) 1979

Cuyo patrón es: **D(h6), Tf8, T(b7), Ph4, rh7, pg7, taboo(g6), 1 ♕h6+**

Nótese que el rey negro no puede ir a g6 por su propio peón. Esto es otra manera de describir la no–disponibilidad de la casilla g6.

6.1.3. Conectores lógicos

Un patrón consiste en definir las piezas en el tablero en ciertas casillas o atacando a alguna(s) casilla(s) en particular. El patrón puede entonces considerarse como un conjunto de definiciones (o instrucciones del lenguaje), conectadas por un el conector lógico "Y" (AND). También podemos definir el conector lógico "O" (OR), el cual es usado cuando una pieza en el patrón pudiese estar en una casilla o en otra. En la mejor tradición del lenguaje Prolog, usamos "," para el AND lógico y ";" para el OR. Pero el lenguaje puede aceptar además de estos símbolos las palabras AND y OR.

Por ejemplo, en la siguiente posición (figura 6.7:

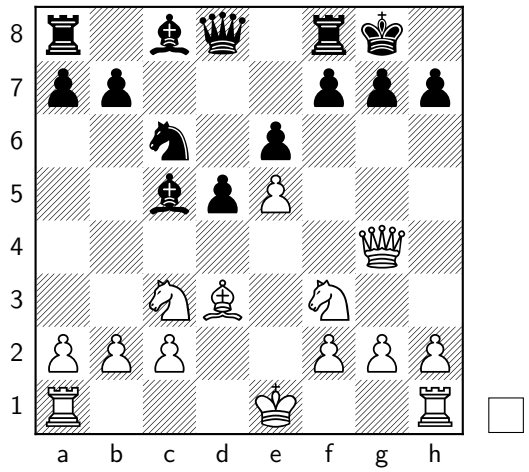

Figura 6.7: **Pestalozzi, M. - Duhm, D**, Bern 1900

El "sacrificio griego" tiene el siguiente patrón:
Ad3, Cf3, Pe5, ph7, pg7, pf7, pe6, rg8
Dd1 ; Dg4 //Notación alternativa es "Dd1 OR Dg4".
1 ♗×h7+
La descripción "Dd1 ; Dg4" indica que la dama blanca podría estar en la casilla d1 *o* en la casilla g4.

6.1.4. Patrones posicionales (*priyomes*)

Además de los factores tácticos en las posiciones, el lenguaje de descripción puede también tratar los patrones posicionales. La mayoría de estos patrones dependen de la estructura de peones para ambos jugadores. Definiendo **structw[lista de peones]** para las blancas y **structb[lista de peones]** para las negras, y el consejo sobre cómo jugar, puede construirse un patrón posicional.

Los patrones de esta naturaleza son más difíciles de definir porque en general no hay un método directo para lograr la ventaja como ocurre en los patrones táctico. Pero de todas maneras hay formas de definir este tipo de patrones[3].

[3]En la literatura ajedrecística los jugadores usan el término "priyome", un sustanti-

Hay muchos *priyomes* populares. Uno de ellos es la explotación de una casilla débil, por ejemplo, la casilla c6. En la partida **Botvinnik–Donner**, Amsterdam, 1963 (ver la figura 6.8), la casilla c6 es débil. Los pasos para obtener la ventaja son:

1. Cambiar los alfiles de casillas blancas
2. Asegurar c6 con un peón blanco en b5
3. Ocupar el agujero con un caballo.

vo ruso que se usa para representar una maniobra típica o una técnica en ajedrez (por ejemplo, un patrón). Un "priyome" típico es el que hemos mencionado antes en la *regla de Tarrasch*, de colocar una torre detrás de los peones, ya sean propios o los del rival.

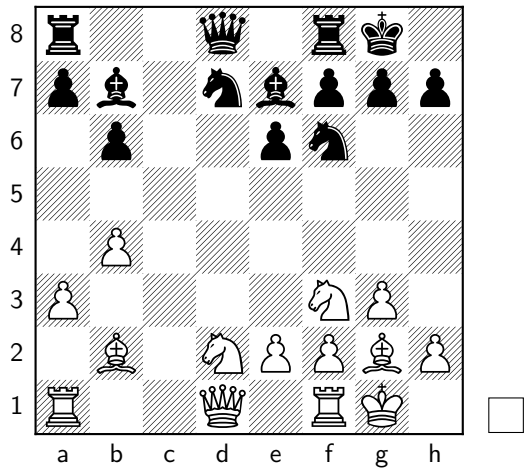

Figura 6.8: **Botvinnik, M - Donner, J.**, Amsterdam 1963

El patrón se puede describir con la ayuda de la conformación de los peones, su estructura: structw(a3, b4, e2, f2, g3, h2),

structb(a7, b6, e6, f7, g7, h7), con el consejo: **hacerse de la casilla c6** para las blancas mediante los pasos mencionados[4].

Un segundo ejemplo del mismo patrón puede verse en Aronian–Carlsen, Elista 2007:

[4]Botvinnik desarrolla el plan con: **14 ♘d4 ♗×g2 15 ♔×g2 ♕c7 16 ♕b3 ♖fc8 17 ♖fc1 ♕b7+ 18 ♕f3 ♘d5 19 e4 ♘5f6 20 b5 a6 21 ♘c6** y las blancas están mucho mejor.

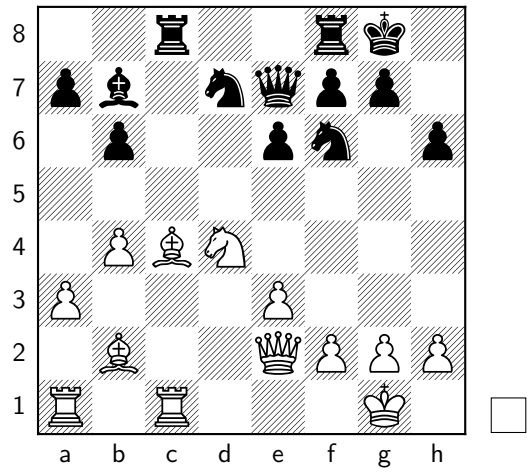

Figura 6.9: **Aronian, L. - Carlsen, M.**, Elista (Candidates) 2007

En este caso el "priyome" es:
structw(a3, b4, e3, f2, g2, h2),
instructb(a7, b6, e6, f7, g7, h6)
con el consejo **hacerse de la casilla c6 para el blanco**[5].

Puede verse que la estructura en ambos ejemplos no es exactamente la misma, pero la casilla "c6" débil es más que obvia. De nuevo, la estructura de peones no tiene que ser exacta, sino similar en algunos aspectos[6].

[5] Aronian sigue los pasos mencionados: **16 ♗a6 ♗×a6 17 ♕×a6 ♖×c1+ 18 ♖×c1 ♘b8 19 ♕c4 ♖d8 20 h3 ♘e8 21 b5!** y eventualmente gana la partida.

[6] En este caso la clave es la debilidad en la casilla c6.

Se puede definir entonces un patrón posicional como una posición en ajedrez que no tiene una conclusión táctica. En este sentido, las siguientes dos posiciones 6.10 y 6.11 son muy ilustrativas.

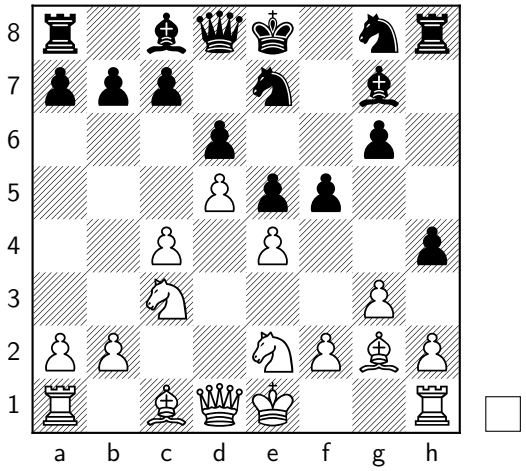

Figura 6.10: **Korchnoi, V. - Short, N.**, Rotterdam 1990

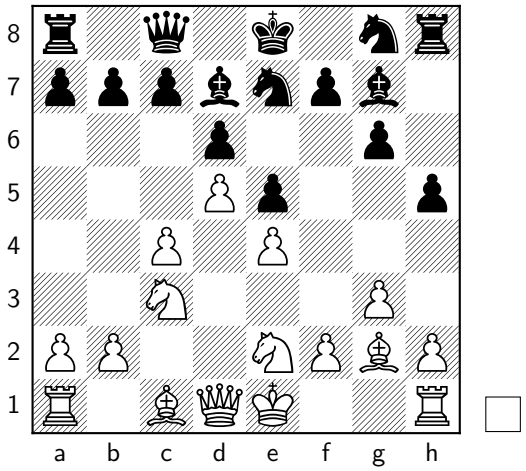

Figura 6.11: **Korchnoi, V. - Polugaevsky, L.**, Leningrad (URS ch) 1963

Korchnoi dice acerca de la siguiente posición (ver siguientes dos diagramas): "Yo jugué una partida contra un joven gran maestro [Nigel Short]. Y de repente, después de 10 jugadas aproximadamente, me di cuenta que había jugado esta posición en 1963 – 27 años antes, contra Polugaevsky) Yo realicé una jugada inusual y muy interesante (**10 ♘g1**) y gané la partida [...] Entonces regresé a casa y miré mi encuentro con Polugaevsky en donde no era definitivamente la misma posición. Y sí, la idea era correcta, pero la posición era diferente" [51].

Korchnoi indica que las posiciones son diferentes, pero no lo son tanto. Ambas tienen el mismo priyome. Por ello, no es de sorprenderse que la jugada en ambas partidas sea la misma.

6.1.5. Resumen del lenguaje de descripción de patrones de ajedrez

Las instrucciones del lenguaje son:

P1(P2) – La Pieza 1 ataca/defiende a la Pieza 2.
P(casilla) – La pieza P ataca una casilla.
P casilla – La pieza P en la casilla.
taboo(casilla) – Define una casilla no disponible al defensor.
structw[lista de peones] – Define la estructura de los peones blancos en el tablero.
structb[lista de peones] – Define la estructura de los peones negros en el tablero.
Conectores lógicos (",", ";") – "," es AND. ";" es OR. Se pueden usar las palabras AND y OR también.
Comentarios – Son por línea y el delimitador es "//", por ejemplo, textit//Este es un comentario.
Las iniciales de las piezas blancas son en mayúsculas **R,D,A,C,T,P)** y las de las piezas negras en minúsculas **(r,d,a,c,t,p)**.

6.1.6. Priyomes importantes

Los siguientes priyomes de ajedrez son algunos de los más conocidos y se basan en el trabajo de Soltis [76].

Priyome #1

Henry Bird, un maestro británico del siglo XIX desarrolla un priyome en donde la estructura de peones pretende ocupar con un caballo la casilla e5, apoyada por un alfil en b2 y otro caballo eventualmente en f3. En la partida Bird–Riemann, DBS Kongress, 1885, se llega a la siguiente posición después de las jugadas: **1 f4 d5 2 ♘f3 c5 3 e3 ♘c6 4 ♗b5 ♗d7 5 O-O e6 6 b3 ♘f6 7 ♗b2 ♗e7 8 ♗×c6 ♗×c6 9 ♘e5 ♕c7 10 d3 ♖g8**

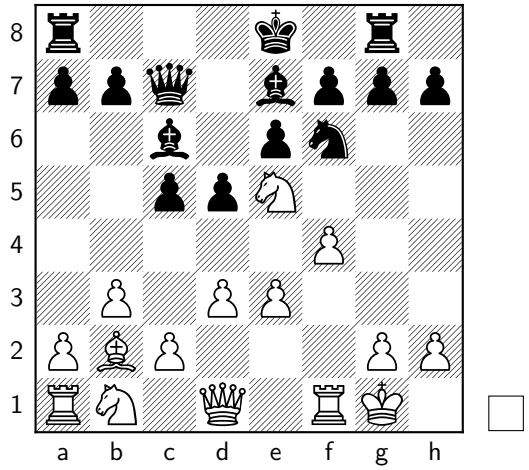

Figura 6.12: **Bird, H. – Riemann, F.**, DBS Kongress 04, 1885

El priyome se basa entonces en apoyar con todo al caballo en e5. El patrón en este caso es:
structw(a2, b3, c2, d3, e3, f4, g2, h2)
structb(a7, b7, c5, d5, e6, f7, g7, h7)
El consejo a seguir es jugar (por parte de las blancas) Cbd2 seguido de Cf3 con ventaja para las blancas.

Hay muchísimas partidas con esta estructura en donde las blancas tienen buenos resultados. Lo interesante de este priyome es que tiene una imagen en espejo, en donde la casilla a ocupar es d5 en lugar de e5. En la partida **Tal –**

Najdorf, Bled 1961, se llegó a la siguiente posición después de las jugadas: **1 e4 c5 2 ♘f3 d6 3 d4 c×d4 4 ♘×d4 ♘f6 5 ♘c3 a6 6 ♗e2 e5 7 ♘b3 ♗e7 8 ♗g5 ♗e6 9 O-O O-O**

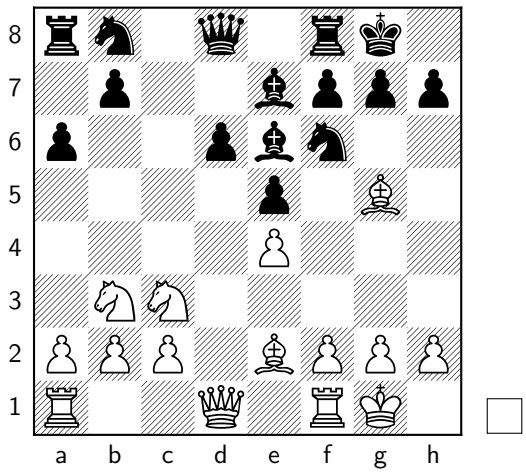

Figura 6.13: **Tal, M.** – **Najdorf, M.**, Bled 1961

Tal sigue con el plan de ocupar la casilla d5 con su caballo con: **10 ♗×f6! ♗×f6 11 ♕d3 ♘c6 12 ♘d5** en donde las blancas ganan eventualmente la partida.

El patrón en este caso es:

structw(a2, b2, c2, e4, f2, g2, h2)
structb(a6, b7, d6, e5, f7, g7, h7)
La recomendación es eliminar el caballo defensor de la casilla d5 y eventualmente ocupar un caballo en esa casilla.

Finalmente el mismo patrón se observa en la partida **Zsofia Polgar** – **Apol, L.**, Reykjavik 1988. (Ver siguiente diagrama): **1 e4 c5 2 ♘f3 d6 3 d4 c×d4 4 ♘×d4 ♘f6 5 ♘c3 a6 6 ♗c4 e6 7 b3 ♗e7 8 O-O ♕c7 9 ♗e3 ♘c6 10 f4 ♗d7 11 ♕f3 ♖c8 12 f5 ♘×d4 13 ♗×d4 e5 14 ♗e3 ♗c6 15 ♕g3 ♖g8**

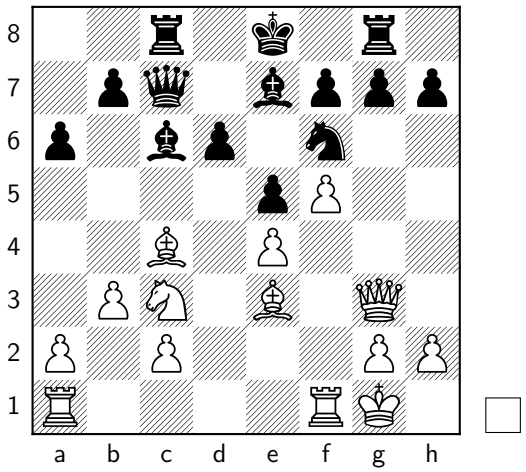

Figura 6.14: **Polgar, Zsofía – Apol, L.**, Reykjavik 1988

El patrón es:
structw(a2, b3, c2, e4, f5, g2, h2)
structb(a6, b7, d6, e5, f7, g7, h7)
Y el consejo es ocupar la casilla g5, eliminando al defensor de f6.

Priyome #2

Muchas veces las negras se expanden en el flanco dama con los peones: ... a6 y ... b5. Esto ocurre en diversas aperturas diferentes, desde el gambito de dama hasta la defensa siciliana. Las blancas tienen sin embargo un priyome claro para desarmar la peligrosa estructura de peones del negro: a2-a4! Por ejemplo, en la partida **Najdorf – Fischer**, Santa Mónica 1966, que comienza de la siguiente manera: **1 d4 ♘f6 2 c4 g6 3 ♘c3 ♗g7 4 e4 d6 5 ♗e2 O-O 6 ♗g5 c5 7 d5 e6 8 ♘f3 h6 9 ♗h4 e×d5 10 c×d5 g5 11 ♗g3 b5 12 ♘d2 a6 13 O-O ♖e8 14 ♕c2 ♕e7 15 ♖ae1 ♘bd7**

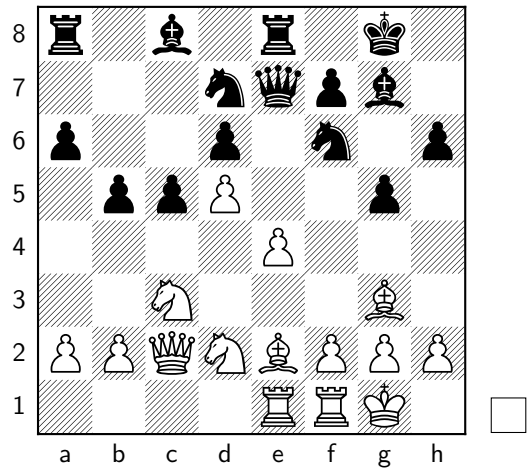

Figura 6.15: **Najdorf, M – Fischer, R.**, Santa Monica 1966

El patrón asociado es:
structw(a2, b2, d5, e4, f2, g2, h2)
structb(a6, b5, c5, d6, f7, g5, h6)
y el consejo a seguir es atacar la estructura de peones negros con a2-a4.

Otro ejemplo del ataque a la expansión negra de peones en el flanco dama puede verse en la partida **Euwe – Sanguinetti**, Mar del Plata 1947, que empezó así: **1 d4 d5 2 c4 e6 3 ♘c3 ♘f6 4 ♗g5 ♘bd7 5 e3 ♗e7 6 ♘f3 O-O 7 ♖c1 c6 8 a3 a6 9 ♗d3 b5 10 c×d5 c×d5 11 ♘e5 ♗b7 12 f4 ♘e8 13 O-O f5 14 ♗×e7 ♕×e7**

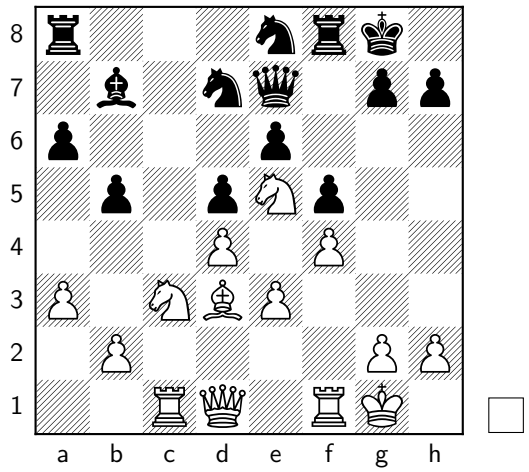

Figura 6.16: **Euwe, M. – Sanguinetti, R.**, Mar del Plata 1947

El patrón asociado a este priyome es:
structw(a3, b2, d4, e3, f4, g2, h2)
structb(a6, b5, d5, e6, f5, g7, h7)

El consejo es atacar la estructura expansiva de los peones del negro con a4.

La partida continúa con: **15 a4! b4 16 ♘e2 ♘df6 17 a5 ♘d6 18 ♕a4 ♘de4 19 ♖c2 ♖ac8** y las blancas a la postre ganan. El problema es que **15 a4** obliga al rival a dejar casillas débiles fundamentales, motivo común en la derrota.

Priyome #3

Uno de los priyomes más populares se da en la Defensa Siciliana, variante Najdorf. La estructura es común en muchas de las estructuras de peones de la defensa siciliana. La torre negra en c8 amenaza al caballo blanco de c3. El patrón en este caso puede representarse así:

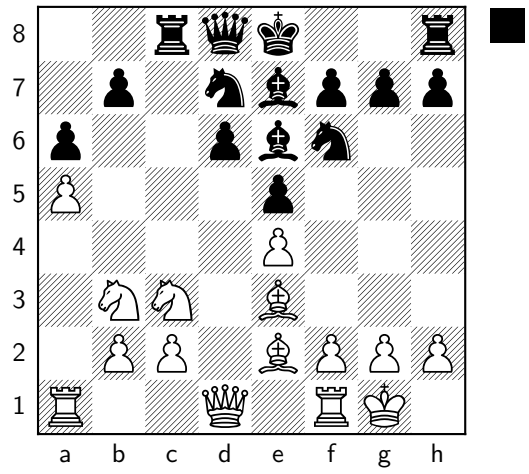

Figura 6.17: **Kovchan, A.** – **Nepomniachtchi, I.**, Rusia (ch) 2010

structw(a5, b2, c2, e4, f2, g2, h2)
structb(a6, b7, d6, e5, f7, g7, h7)

Y el consejo a seguir es (para las negras) sacrificar la calidad (torre vs caballo o alfil) pero capturando entonces el peón de e4 blanco indefenso. Además, la estructura de peones blancos en el flanco dama queda muy débil. Hay una plena compensación por la calidad entregada.

Por ejemplo, en la partida **Kovchan** – **Nepomniachtchi**, URSS 2010, (ver diagrama 6.17), las negras (en la posición del diagrama), juegan: **11. . . ♖×c3**, eliminando al caballo defensor del peón blanco de e4. Después de **12 b×c3 ♘×e4 13 ♕d3 f5 14 f4 ♕c7** las negras logran hacerse de la ventaja y finalmente ganan la partida[7].

[7]La partida completa es así: **1 e4 c5 2 ♘f3 d6 3 d4 c×d4 4 ♘×d4 ♘f6 5 ♘c3 a6 6 ♗e3 e5 7 ♘b3 ♗e7 8 ♗e2 ♗e6 9 O-O ♘bd7 10 a4 ♖c8 11 a5 ♖×c3 12 b×c3 ♘×e4 13 ♕d3 f5 14 f4 ♕c7 15 c4 O-O 16 c5 d×c5 17 ♘d2 c4 18 ♘×c4 ♖d8 19 ♖ab1 ♘df6 20 ♕b3 ♘d5 21 ♘×e5 ♗c5 22 ♗×c5 ♕×c5+ 23 ♔h1 ♘×f4 24 ♗c4 ♕×e5 25 ♖×f4 ♕×f4 26 ♗×e6+ ♔h8 27 ♕f3 ♕h6 28 ♗a2 ♘d2 29 ♕d1 ♕h4 30 ♖×b7 ♘f3 31 ♕×d8+ ♕×d8 32 g×f3 ♕d1+ 33 ♔g2 ♕×c2+ 34 ♔h3 h5 35 ♗f7 g5 36 ♗×h5 ♕f2 37 ♖b6 ♕h4+ 38 ♔g2 ♕×h5 39 ♖×a6 g4 40 ♖a8+ ♔g7 0–1**.

Priyome #4

Un caballo apoyado por dos peones dentro del territorio enemigo los rusos lo denominan "un anillo". Por ejemplo, en una partida de 1783, **Von Bruehl – Philidor**, en Londres, las negras ubican un caballo en la posición de anillo (en este caso la casilla c4), haciéndose de la ventaja:

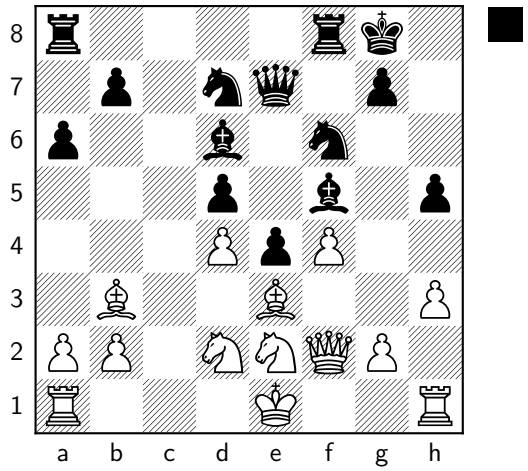

Figura 6.18: **Von Bruehl, H. – Philidor, F.**, Londres 1783

El patrón correspondiente es:
structw(a2, b2, d4, f4, g2, h3)
structb(a6, b7, d5, e4, g7, h5)
con el consejo de jugar **...b5**, **...♘b6** y **...♘c4**, armando el *anillo* (también conocido como *anillo de Philidor*).

La partida continúa: **17...b5 18 O-O ♘b6! 19 ♘g3 g6 20 ♖ac1 ♘c4** y las negras se hacen de la ventaja, la cual capitalizan muchas jugadas después.

Un esquema parecido en donde el priyome se observa de nuevo es cuando en la partida se intercambian damas en la casilla b3, como por ejemplo, en **Janowski** – **Marshall**, de su match de 1905 (ver la figura 6.19).

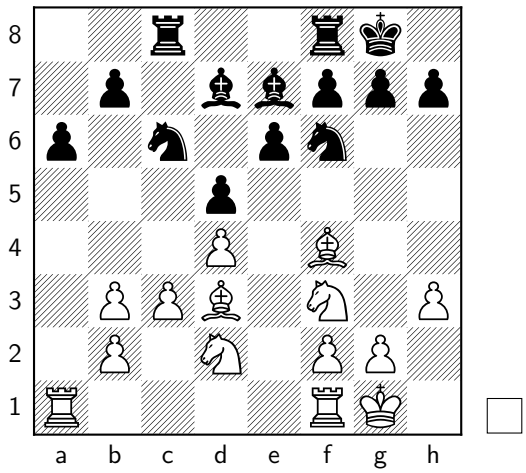

Figura 6.19: **Janowsky, D.** – **Marshall, F.**, Match 1905

El patrón asociado es:
structw(b2, b3, c3, d4, f2, g2, h3),
structb(a6, b7, d5, e6, f7, g7, h7)
con el consejo de ocupar la casilla c5 después de jugar primero **b4**, ♘**b3** y ♘**c5**, con ventaja del blanco. Nótese que no se puede impedir que las blancas realicen el plan jugando **. . . . .b6** porque se perdería el peón de a6 negro.

6.2. Ventajas del lenguaje de descripción de posiciones

Una abstracción visual tiene ventajas muy claras en diversos dominios. Por ejemplo, Alexander afirma que un patrón (de diseño) –si puede ser dibujado– es decir, si se puede hacer un esquema gráfico del mismo, entonces estamos ante la esencia del patrón [2]. Sin embargo, en otros dominios –por ejemplo las matemáticas– esto no es necesariamente cierto. La expresión $y = x^2$ representa la gráfica de una parábola centrada en el cero. El

poder de la abstracción en la definición de un lenguaje particular, en donde las matemáticas son un estupendo ejemplo, parece evidente[8].

En el lenguaje de descripción de posiciones de ajedrez definido en el capítulo 5, podemos definir de manera mucho más abstracta las configuraciones de las piezas, sin que necesariamente se tengan que definir las posiciones que ocupan las respectivas piezas en el patrón. Por ejemplo, en la siguiente posición podemos definir el ataque de un alfil blanco sobre el peón negro de *h7*.

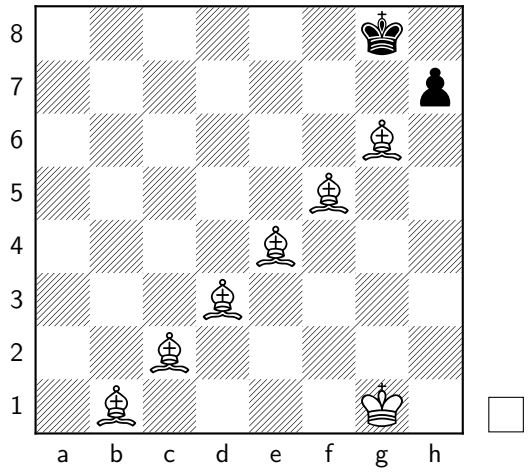

Figura 6.20: **Representación visual del patrón A(ph7)**

[8]https://www.dpmms.cam.ac.uk/ wtg10/grammar.pdf

En una sola expresión, "A(ph7)", definimos que el alfil blanco puede estar en cualquiera de las siguientes casillas: b1, c2, d3, e4, f5 y g6. Esto además, abrevia la necesidad de hacer un diagrama para cada una de las posibilidades mencionadas.

Se pueden definir patrones más complejos, por ejemplo, encadenando líneas que definen las características del patrón. Se asume que en este encadenamiento se usa por omisión el conector "AND". En este caso definimos el patrón de la famosa partida entre **Lasker vs. Bauer**, Amsterdam 1889 (figura 6.21).

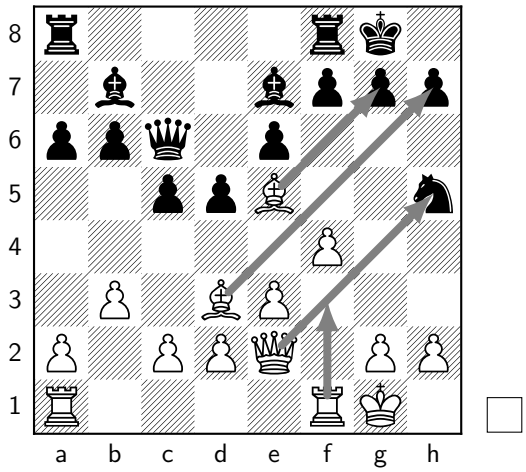

Figura 6.21: **Lasker - Bauer**, Amsterdam 1889

El patrón: **D(h5), A(ph7), A(pg7), T(f3) [ó T(c3)], 1 ♗×h7+** se repite casi 100 años después en la partida de **Browne, W. - Miles, A.**, Lucerna (ol) 1982 (figura 6.22).

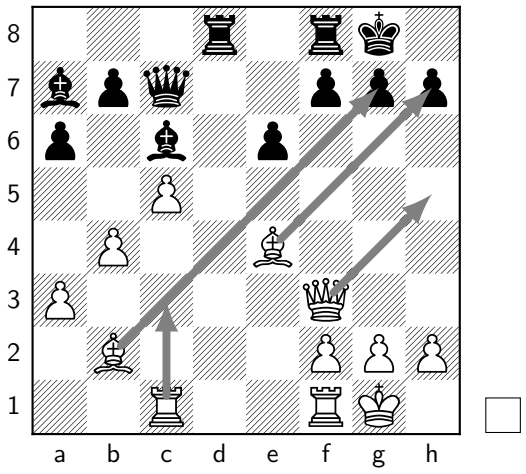

Figura 6.22: **Browne, W. - Miles, A.**, Lucerna (ol) 1982

6.2.1. Valoración de los patrones encontrados

Los ataques y las defensas tienen valores diferentes. Por ejemplo, no es igual que una dama esté atacando a un peón a que un peón ataque a la dama. (véase figura 6.23). Así, un peón negro que ataca la dama enemiga tiene un mayor valor que una dama blanca que ataca a un peón del oponente. La razón es simple: la dama vale el equivalente a 9 peones.

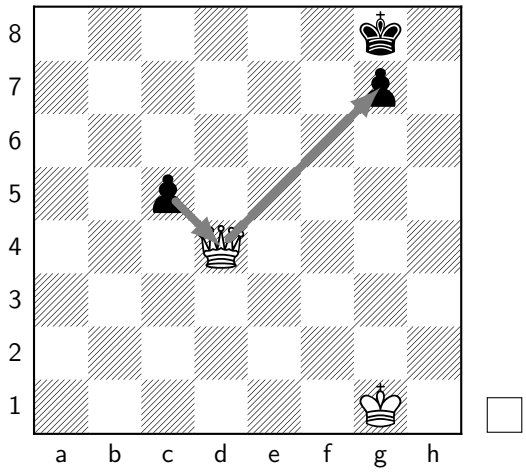

Figura 6.23: **Un ejemplo de la diferente valoración de los ataques y las defensas**

Por ello es necesario otorgar cierto peso al tipo de patrones que se encuentren, porque esto depende de la fuerza intrínseca de las piezas. De esta manera el lenguaje de descripción de patrones tiene que tener una manera de valorar toda la configuración completa basada en los valores que dan los ataques y defensas. Una vez definidos todas estas configuraciones, se puede pasar al siguiente paso.

6.2.2. Búsqueda de los patrones definidos, en una base de datos de partidas

Dados el conjunto de patrones primitivos, el procedimiento a realizar es una búsqueda sobre la base de datos que registra prácticamente todo el historial de partidas de ajedrez jugadas en todo el mundo desde el año 1560

aproximadamente, hasta nuestros días[9]. Esto es equivalente a unos 7 millones de partidas jugadas en competencias oficiales. La búsqueda de todos estos patrones puede resumirse en revisar qué posiciones contienen los patrones de interés. A pesar de la simplicidad del mismo, 7 millones de partidas equivale a analizar en promedio 560 millones de posiciones diferentes (una partida promedio es de unos 80 movimientos en total, 40 de blancas y 40 de negras). Un programa que calcule todos los patrones de interés podría llevar mucho tiempo (asumiendo que se tiene un conjunto finito de ellos ya establecidos). Una posibilidad interesante es reducir la base de partidas a encuentros de un nivel de ajedrez de primera línea, es decir, partidas entre grandes maestros, lo que podría reducir el espacio de búsqueda significativamente, sin detrimento de los patrones ganadores encontrados[10].

6.2.3. Resolución de estos patrones ganadores

Para corroborar que se habla de patrones ganadores (o bien, que dan ventaja considerable), es posible basarse en los motores de ajedrez, en los programas actuales que juegan a nivel de Gran Maestro sin dificultades[11]. Actualmente los motores de ajedrez rebasan a los mejores exponentes del ajedrez y además, hay una manera de medir el nivel de esos programas, basados en la llamada *Prueba Bratko–Kopec*[12].

Hay preguntas por resolver, por ejemplo, ¿Se puede asegurar que nuestros patrones primitivos forman un conjunto exhaustivo de relaciones ganadoras? ¿cuál es el conjunto mínimo de condiciones que cumplen con un patrón ganador? ¿Hay una esencia sobre los atributos que definen esa característica

[9]Para ello se cuenta con la *MegaBase 2018* de la empresa alemana Chessbase, que cada año genera una nueva base de partidas incluyendo las que se jugaron en el último año.

[10]Los grandes maestros son los mejores exponentes del ajedrez. Tiene lógica buscar los patrones en partidas entre ellos.

[11]Se pueden usar los motores de ajedrez como *Houdini* o *Komodo 10*, dos programas comerciales, los cuales pueden valorar cualquier posición con un fuerte grado de certidumbre. Se puede entonces comparar las valoraciones del motor contra el patrón definido para observar si es ganador o no.

[12]Diseñada por Ivan Bratko y Danny Kopec en 1982 para evaluar la habilidad de las máquinas (y seres humanos) para jugar bien al ajedrez. Basándose en la presencia o ausencia de cierto conocimiento (experto, maestro, principiante). Esta prueba es un estándar que tiene más de 30 años de usarse en ajedrez y la experiencia ha mostrado que es una de las pruebas más confiables para valorar el nivel de máquinas y seres humanos. http://kopecchess.com/bratko-kopec-test/

ganadora en la configuración encontrada? Estas preguntas pueden resolverse en la medida que definamos el conjunto de configuraciones ganadoras mínimas.

6.3. Resumen

En este capítulo se revisa el caso de estudio, el juego del ajedrez, en el cual se plantea la alternativa de un lenguaje de descripción de patrones, el cual puede generalizar las configuraciones ganadoras que puedan presentarse en las diferentes partidas de ajedrez. El lenguaje de descripción es lo suficientemente simple y se dan ejemplos de su aplicación. Finalmente se aumentan las posibilidades del lenguaje de descripciones de forma que puedan definirse los "priyomes". Se dan ejemplos y se categorizan los mismos al final del capítulo. Por último, se valoran los patrones definidos así como su alcance.

Capítulo 7

Formalismo del lenguaje de descripción de patrones

En este capítulo se describe el lenguaje de patrones en una descripción algebraica formal. Se busca desarrollar un modelo algebraico para la descripción de las posiciones de ajedrez. En este marco, los objetos ajedrecísticos consisten en primitivas como las piezas, el tablero de 64 casillas, así como los movimientos (ataques y defensas) de las diferentes piezas, además de las operaciones que transforman los objetos ajedrecísticos en las diferentes posiciones que pueden encontrarse. Se debe tener operaciones que puedan ser combinadas para generar incluso macro-operaciones. Todo esto puede hacerse en el marco del álgebra de conjuntos, de forma que la descripción de las posiciones en ajedrez pueda ser claro y preciso. El utilizar esta álgebra particular permite comprender las propiedades de las posiciones en ajedrez, por ejemplo, si son conmutativas, asociativas y distributivas -considerando los diferentes operadores. De esta forma surge una descripción matemática para el juego del ajedrez. En principio este resultado podría expandirse a otros juegos de suma-cero e información perfecta.

La notación ajedrecística (llamada notación algebraica –véase el Apéndice 1), permite el poder reproducir las partidas de los jugadores de manera clara e inequívoca. Sin embargo, esta notación no puede capturar (y tampoco se crea con ese fin), lo que ocurre en las posiciones en ajedrez. Es decir, la notación se limita a poner las piezas en las casillas correspondientes, de acuerdo con los movimientos registrados en la partida, pero no puede dar ninguna información sobre el estado de la partida, es decir, quién tiene la ventaja, quién está mejor, qué jugada es la más adecuada para hacer en este

momento, etcétera.

Un juego de suma–cero e información perfecta puede verse como la aplicación de funciones sobre un conjunto de objetos (que son el juego mismo). Por ejemplo, en el caso del juego del NIM, los movimientos de un jugador pueden considerarse funciones que se aplican a la posición actual del juego. Lo mismo puede decirse con juegos como el ajedrez, en donde los movimientos de los jugadores están dictados por una función de evaluación que –en principio– le indican a cada jugador cómo debe actuar.

7.1. Hacia un álgebra de conjuntos para los patrones en ajedrez

Un conjunto es una colección de objetos llamados elementos. Un elemento puede ser cualquier cosa, números, funciones, líneas, etcétera. Un conjunto es un objeto que puede contener muchos elementos [64].

Podemos definir, para el caso del ajedrez, lo siguiente [39]:

1. S_1, S_2, ..., S_m como posiciones legales en el tablero de ajedrez, las cuales podemos enumerar arbitrariamente pero en una secuencia ordenada. Así, S_m y S_{m+1} pueden considerarse dos configuraciones de piezas en el tablero que son distintas si en ambas el turno no es del mismo jugador. De hecho, consideramos S_1 como la posición inicial del ajedrez.

2. La relación $S_q \rightarrow S_p$ es posible si las reglas del ajedrez permiten la transición desde S_q a S_p, esto es, S_p puede llegar en una jugada a partir de S_q.

3. Dada una S_p hay un número de configuraciones (con $i \geq 0$) S_{p_v} para las cuales $S_p \rightarrow S_{p_v}$ con $v = 1, ..., i < 150$ [1] [2]. A estas las llamamos

[1] Aparentemente Euwe considera 150 movimientos como el límite máximo de una partida de ajedrez.

[2] Aunque hay diferentes cálculos al respecto, aparentemente hay 5898 posibles movimientos antes de que se decrete un empate (de acuerdo a las reglas vigentes del ajedrez). Para el cálculo se tienen: 2 lados (blancas y negras) * (6 movimientos de peones que pueden coronar * 8 peones + 8 capturas de los peones coronados + 3 capturas de otras piezas opositoras que obligan a tener cuatro para ser capturadas por los peones) * la regla de las 50 jugadas (para otorgar el empate) - 4 veces el lado que hace que un peón se mueva o bien haga una captura) * (1/2 de movimientos) = 5898

configuraciones para las cuales $i = 0$ no se tiene elección (rey ahogado o jaque mate). A esto le denominamos *elección libre* i [3].

4. S es el conjunto de todas las posibles partidas de ajedrez.

S_1 es la configuración inicial del tablero de ajedrez. A partir de ahí se pueden generar un número arbitrario de S_p. Si $p \leq$ *elección libre* a partir de la *elección libre*, entonces la p-ésima de todas las configuraciones alcanzables a partir de S_α se generan. El ordenamiento se da por el índice. Si $p >$ *elección libre* a partir de S (*elección libre* $= 0$), entonces no se genera nada. Cabe decir que esta definición excluye todas las partidas de ajedrez que no terminan en ahogado o jaque mate. Desde luego hay una regla más a tomar en cuenta, que es la regla de las 50 jugadas sin movimiento de peón o captura, que lleva al empate, pero que se analiza más adelante.

Esta es la manera de crear las ramas del árbol (variantes) a partir de la posición S_1. El número de ramas en cada configuración es igual a la variable i (*elección libre*). Cabe señalar que S tiene un subconjunto E que es el de un número finito de partidas de ajedrez.

7.1.1. Definición de los patrones en términos de conjuntos

Los patrones son subconjuntos de posiciones en donde ciertas condiciones ocurren, esto es, indicar qué piezas atacan, cuáles defienden y las piezas que son parte del patrón. Podemos definirlos (en el lenguaje de descripción) de la siguiente manera:

$$P_m = \{a_1, a_2, ..., a_n, d_1, d_2, ..., d_k, f_1, f_2, ..., f_{32}\}$$

Sea $S = \{S_1, S_2, ..., S_n\}$ las posibles posiciones legales de una partida de ajedrez.

(https://www.chess.com/blog/kurtgodden/the-longest-possible-chess-game). Pero este valor es muy grande y en la práctica hay muy pocas partidas que hayan pasado de 260 jugadas.

[3]Cuando $i = 0$ lo que ocurre es que no hay movimiento posible y por eso no hay *libre elección*.

Podemos definir $S_{1_1} = \left\{\sum_{i=1}^{n} a_i, \sum_{k=1}^{m} d_k, \sum_{j=1}^{32} f_j\right\}$ como todos los ataques (a_i) y defensas (d_k) de las piezas en el tablero, además de las posiciones que ocupan esas piezas (f_j).

Cuando $S_{1_n} \cap P_m \neq \varnothing$ entonces existe al menos un patrón P_i y puede procesarse.

7.1.2. Ejemplo en el ajedrez

El patrón del sacrificio griego se define de la siguiente manera(figura 7.1).

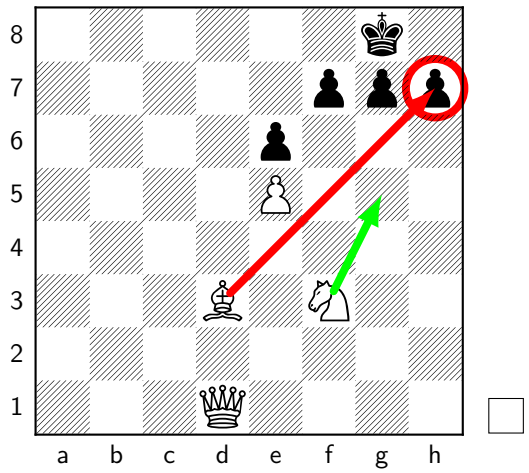

Figura 7.1: Ejemplo del sacrificio griego.

Ad3, Cf3, Pe5, ph7, pg7, pf7, pe6, rg8
Dd1 ; Dg4 //Notación alternativa es "Dd1 OR Dg4".
1 ♗×h7+ La descripción "Dd1 ; Dg4" indica que la dama blanca podría estar en la casilla d1 *o* en la casilla g4.

Procesando este patrón hallamos que se ha dado en unas 1000 partidas (sobre una base de más de 7 millones de las mismas). El software escrito para tal fin nos entrega los encuentros donde se da este popular sacrificio de pieza. Un par de ejemplos (figuras 7.2 y 7.3).

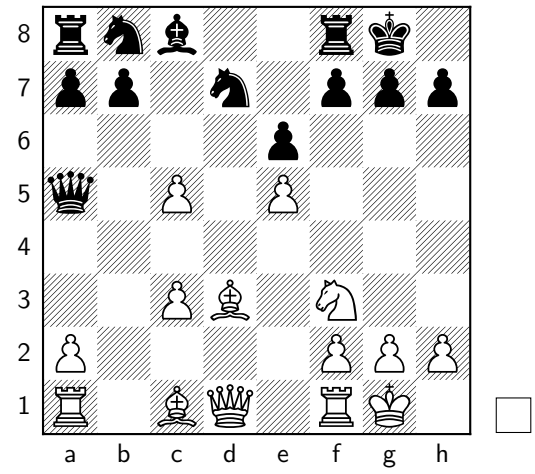

Figura 7.2: **Herrmann, Hans - Harms, Otto**, Lueneburg 1947.

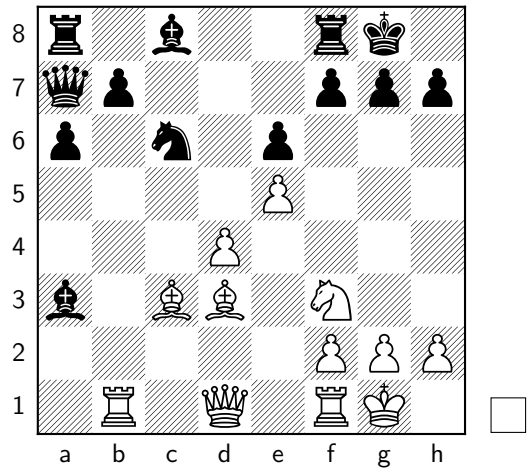

Figura 7.3: **Pirc, Vasja - Porreca, Giorgio**, Bled 1963

El análisis completo del sacrificio griego puede verse en el capítulo 4.5.3.

7.1.3. Ejemplo en el juego del gato

El mismo esquema de la teoría de conjuntos puede usarse en el caso del juego del gato, en donde aquí incluso la dificultad es menor porque este juego puede ser resuelto definitivamente, es decir, se sabe siempre el resultado final del mismo –dependiendo de cómo los jugadores jueguen.

Podemos representar con $S = \{S_1, S_2, ..., S_n\}$ las posibles posiciones legales de todas las partidas en el gato. Estas pueden representarse usando la notación simbólica, a manera de lista, por ejemplo, la posición siguiente (ver figura 5.15) es ganadora para el jugador que lleva las "X": [x, o, b, b, b, b, b, b, b], donde la "b" representa una casilla vacía.

Un patrón ganador en el juego puede representarse como:

$$P = \{x, o, b, b, b, b, b, b, b\}$$

.

Aquí de nuevo, la intersección nos indica si existe un patrón o no: Si $S_i \cap P_m \neq \varnothing$ entonces existe al menos un patrón ganador que puede procesarse. Esto es, después del primer movimiento de cada jugador: X1 $\rightarrow$ O2 podemos seguir con X5 $\rightarrow$ O8 $\rightarrow$ X7 que lleva a un patrón ganador.

7.1.4. Propiedades de los conjuntos de patrones

En 1977 John Backus (de IBM), diseña el FP (*Programación funcional*), que describe en su conferencia cuando recibe el Premio Turing en 1977 [9], aunque el premio le fue otorgado por su otras contribuciones a la informática –Fortran y BNF. Backus señala que, en general, los programadores tienden a manipular datos en lugar de funciones, comenzando con los valores de entrada de un programa, que pasan por una serie de funciones hasta que se alcanzan los datos de salida. En la FP, las funciones -que son las primitivas del lenguaje- se combinan de tal manera que producen una función final, que es un programa de computadora. Esto luego se aplica a los datos de entrada y produce una salida, donde no se requieren variables [35]. Este en esencia es un lenguaje funcional, como Prolog, Lisp o Haskell, entre otros.

Las propiedades de esta álgebra de los conjuntos de patrones en ajedrez (en particular), pueden resumirse así [**?**]:

- **Composición** - escrito como $f \circ g$. Dadas dos funciones, f y g, $f \circ g$ es la función obtenida aplicando g al argumento de la función y tomando el resultado de esta función como el argumento f.

- **Construcción** - escrito como $[f_1, f_2, ..., f_n]$. Se crea una secuencia de n elementos donde el i-ésimo elemento se obtiene aplicando f_i a los datos de entrada.

- **Condicional** - escrito como $p \rightarrow f; g$. Si el predicado p es verdadero, entonces se aplica f al argumento y si p es falso, se aplica g.

- **Aplicar a todos** - escrito como αf. Crea una secuencia de la misma longitud que la secuencia de entrada, aplicando f a cada elemento de los datos de entrada.

- **Insertar** - escrito como $/f$. Aplica f a la secuencia formada por el primer elemento de los datos de entrada, seguida de $/f$ para el resto de la secuencia de entrada.

7.2. La forma BNF para el lenguaje de descripción de patrones

La notación FEN (*Forsyth-Edwards Notation*) describe las posiciones de ajedrez de manera precisa. Es una cadena de símbolos ASCII inventada por David Forsyth en el siglo XIX. Sin embargo, es Steven Edwards quien especifica esta notación FEN para las aplicaciones de ajedrez, como parte de la notación portable de partidas de ajedrez llamado PGN (*Portable Game Notation*[4].

Por ejemplo, una posición en ajedrez puede representarse en FEN así (figura 7.4).

[4]https://www.thechessdrum.net/PGN_Reference.txt

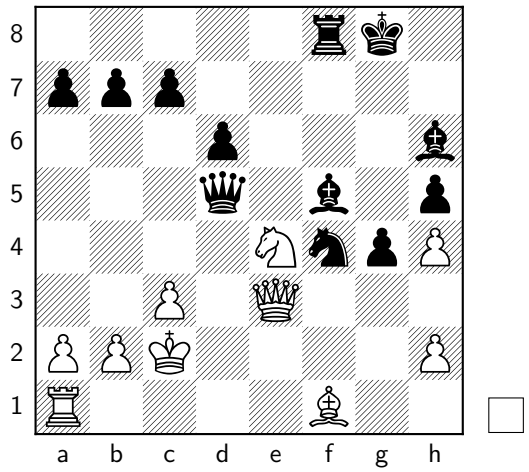

Figura 7.4: Un ejemplo FEN:
5rk1/ppp5/3p3b/3q1b1p/4NnpP/2P1Q3/PPK4P/R4B2 w - - 0 25

El lenguaje de descripción de patrones puede expresarse en notación BNF (*Backus-Naur form*), el cual es un metalenguaje usado para expresar gramáticas libres de contexto, es decir, hablamos de expresar de una manera formal los lenguajes [55].

BNF es muy usado para describir las gramáticas de los lenguajes de programación y de algunos sistemas basados en comandos. El lenguaje de descripción de patrones diseñado puede expresarse de esta manera. Sin embargo, antes hay que expresar la notación FEN, que la usamos para describir y hallar los patrones.

7.2.1. Sintaxis de FEN

Una cadena FEN consiste en seis campos separados por un espacio.

```
<FEN> ::=  <Posición de las piezas>
' ' <Jugador que mueve>
' ' <Posibilidad de enrocar>
' ' <Casilla objetivo en la captura al paso>
' ' <Número de jugadas medias>
' ' <Contador total de movimientos>
```

7.2.2. Posición de las piezas

La posición de las piezas se determina de fila en fila (*rank*), empezando de la octava a la primera fila. Cada fila está separada por el símbolo "/". Las casillas no ocupadas se cuentan de forma entera y las piezas se identifican por su inicial en inglés: K(ing), Q(ueen), B(ishop), kN(ight), R(ook). Se usan mayúsculas para las piezas blancas y minúsculas para las piezas negras.

```
<Posición de las piezas> ::= <rank8>'/'<rank7>'/'<rank6>'/'<rank5>'/'
                             <rank4>'/'<rank3>'/'<rank2>'/'<rank1>
<ranki>       ::= [<digito17>]<pieza>
                  {[<digito17>]<pieza>} [<digito17>] | '8'
<pieza>       ::= <white (piezas blancas)> | <black (piezas negras)>
<digito17>     ::= '1' | '2' | '3' | '4' | '5' | '6' | '7'
<white (piezas blancas)> ::= 'P' | 'N' | 'B' | 'R' | 'Q' | 'K'
<black (piezas negras)> ::= 'p' | 'n' | 'b' | 'r' | 'q' | 'k'
```

7.2.3. Jugador que mueve

Solamente puede ser blancas ("w") o negras ("b").
Así, entonces:

```
<Jugador que mueve> ::= {'w' | 'b'}
```

7.2.4. Posibilidad de enrocar

Si ningún jugador puede enrocarse, se usa el símbolo "-". En cambio, si se tiene el derecho al enroque, si es corto, se anota "K" (para blancas) y "k" (para negras). Similarmente se usa "Q" para el enroque largo (blancas) y "q" para el respectivo enroque de las negras.

```
<Posibilidad de enrocar> ::= '-' | ['K'] ['Q'] ['k'] ['q'] (1..4)
```

7.2.5. Casilla objetivo en la captura al paso

La casilla objetivo *al paso* se especifica cuando un peón se mueve dos casillas, sin importar que pueda existir la eventualidad de la captura al paso.

```
<Casilla objetivo en la captura al paso> ::= '-' | <epsquare>
<epsquare>   ::= <Letra de la fila> <eprank>
<Letra de la fila> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h'
<eprank>     ::= '3' | '6'
```

7.2.6. Número de jugadas medias

El número de jugadas medias es un número decimal de la cantidad de medias jugadas requeridas para la regla del empate de las cincuenta jugadas. Se pone en cero cada vez que hay una captura o un movimiento de peón. De otra forma se incrementa.

```
<Número de jugadas medias> ::= <digito> {<digito>}
<digito> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

7.2.7. Contador total de movimientos

Lleva el registro del número total de movimientos en una partida, empezando en la unidad y se incrementa después de cada movimiento de las negras.

```
<Contador total de movimientos> ::= <digito19> {<digito>}
<digito19> ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<digito>   ::= '0' | <digito19>
```

7.3. Descripción del lenguaje de patrones

El lenguaje de descripción de patrones se establece a partir de las siguientes instrucciones [60]:

- **A(B)** – La pieza A ataca/defiende a la pieza B.
- **A(casilla)** – La pieza A ataca una casilla.
- **A casilla** – La pieza A está en una casilla.
- **taboo(casilla)** – Pone como no–disponible a la casilla para el defensor.
- **action(movimiento)** – El movimiento recomendado para el patrón.

- **structw([lista de peones])** – define la estructura de los peones blancos en el tablero.

- **structb([lista de peones])** – define la estructura de los peones negros en el tablero.

- **Conectores lógicos (“,”, “;”)** – “,” es AND. “;” es OR. Pueden usarse alternativamente lasa palabras “AND” y “OR”.

- **Línea de comentarios** – Usando el delimitador “//”. Por ejemplo *//Este es un comentario.*

Las piezas en mayúsculas definen a las blancas (**K,Q,B,N,R,P**) y las minúsculas definene a las piezas negras (**k,q,b,n,r,p**)[5].

7.4. Análisis de los datos experimentales

Se contruye un programa de computadora para encontrar los patrones de ajedrez a partir del lenguaje de descripción. Usamos la Megabase 2018 de la empresa Chessbase en el formato de notación PGN. Esta es una colección de 7,—187,893 partidas jugadas entre 1475 y 2017. Como ejemplo buscamos el patrón del regalo griego (para las blancas), el cual se define como kg8, pf7, pg7, B(ph7), Nf3, Qd1 y Pe5.

Encontramos 1195 partidas en donde la posibilidad del patrón regalo griego está presente:

1. 99 partidas terminan en empate (tablas), 16 %

2. Las blancas ganan en 907 partidas (76 %) (80 % - triunfos y empates)

3. Las negras ganan en 188 partidas (16 %) (20 % - triunfos y empates)

4. En 700 partidas, las blancas tienen un desempeño equivalente a 2043 puntos Elo, con un promedio de 1971 puntos Elo.

5. En 840 partidas las negras tienen un desempeño equivalente a 1708 puntos Elo, con un promedio de 1823 puntos Elo.

Figura 7.5: El software experimental de búsqueda de patrones

Una primera conclusión muestra la efectividad del patrón del sacrificio griego en un 80 % de los juegos de la muestra. Esto significa que podemos guiar la búsqueda de las mejores jugadas usando el patrón en lugar de analizar todas las posibles variantes (con una profundidad determinada), en una posición específica.

La aplicación puede encontrar un patrón descrito por el lenguaje de descripción. Estos patrones pueden escribirse usando un editor de textos ASCII (sin símbolos de control). La aplicación trabaja bajo la plataforma Windows (en sus versiones 7 y 10) y el programa hace uso intensivo del CPU porque

[5]Para una descripción más completa del lenguaje de patrones, véase la sección 6.1.2

se debe hacer la búsqueda completa de cada patrón en cada posición de cada partida.

Se prueba el programa en dos computadoras. La primera usa un procesador AMD FX(tm)-8350 de 8 núcleos, corriendo a 4.0 Ghz, con 4 GB de RAM y Windows 7 Pro (64 bits); y en una segunda computadora con un procesador AMD Ryzen 5 PRO 1600 de seis núcleos, a 3.20 GHz y con 16 GB de RAM, corriendo Windows 10 Pro (64 bits). En la versión más rápida (Windows 10), el software puede encontrar un patrón (con ciertas optimizaciones hechas), analizando todas las posiciones generadas en una partida y comparándolas con el patrón objetivo, en cerca de un segundo por partida. Por supuesto que esto es extremadamente lento, pero no se está usando un sistema de base de datos específico, sino que se parte de las partidas en formato PGN. Este software es simplemente una prueba de concepto.

Sin embargo, se busca optimizar los resultados convirtiendo la base de partidas PGN al formato Forsyth. En [60], se muestra que –mediante un software experimental– se puede buscar un patrón en cerca de 80 posiciones (una partida promedio contiene 40 jugadas, es decir, 80 movimientos). En un intento por acelerar las búsquedas, tomamos la colección de 1030 partidas en donde se encuentra el sacrificio griego y convertimos cada partida en una colección de posiciones en formato FEN. Para ello usamos el programa *pgn2fen*, que convierte archivos PGN a FEN/EPD, escrito por Tim Foden (2002). Este software genera todas las posiciones de un archivo PGN. Por otra parte, convertimos el patrón regalo griego como una cadena FEN. Se escribe entonces un software en Delphi que hiciese búsquedas regulares (*regex*) para buscar el patrón en cada una de las posiciones, las cuales son alrededor de 110,800. El tiempo de respuesta es de unas 4320 posiciones por segundo, unos 54 juegos completos por segundo.

Sin embargo, usando programación funcional a través de Prolog, el mismo archivo de posiciones FEN/EPD se pone como un predicado en una base de datos/programa, en Prolog, y mediante un par de predicados para comparar listas (patrón versus una posición), la velocidad de búsqueda se incrementa a 15,828 posiciones por segundo, es decir, unos 197 partidas por segundo. La diferencia entre las versiones de Delphi y Prolog es de 366 % en favor de este último lenguaje.

7.5. Resumen

Se define formalmente la notación ajedrecística y se añade la definición algebraica formal del lenguaje de descripción de patrones en ajedrez y se muestran los resultados obtenidos.

Capítulo 8

Conclusiones

En este capítulo se presentan las conclusiones del trabajo realizado, así como el trabajo futuro que podría hacerse en este tema.

8.1. Resumen de la investigación

El trabajo se basa en la investigación de las configuraciones o patrones ventajosos que se dan en los juegos de suma–cero e información perfecta, los cuales nacen a partir de sus características particulares. La búsqueda de patrones ventajosos puede considerarse como una importante alternativa a los esquemas tradicionales como son la matriz de pagos o la búsqueda Minimax [80]. Estas dos últimas alternativas engloban muchos de los juegos "que jugamos" y es claro no son suficientes en muchos casos por la propia naturaleza de los juegos en estudio, es decir, no pueden generar a través de estos mecanismos las soluciones óptimas. Además, hay que reconocer que algunos juegos simplemente no tienen una solución final, es decir, no se pueden enumerar todas las posiciones en donde un jugador obtiene ventaja para ganar o simplemente para no perder.

El trabajo se centra en tomar algunos de los juegos de suma–cero e información perfecta más conocidos e investigar la posibilidad de encontrar los patrones ventajosos en las posiciones que ocurren en dichos juegos. Tomamos como ejemplos el juego del gato y el juego de NIM (ambos resueltos totalmente), así como el juego del ajedrez, que sigue siendo un problema abierto, es decir, no está resuelto.

8.2. La hipótesis y su comprobación

En muchos juegos de suma–cero e información perfecta existen patrones, configuraciones que se repiten con frecuencia. El principal objetivo de esta investigación es identificar los patrones como ventajosos o ganadores, incluso en etapas tempranas del juego.

Básicamente se busca:

- Reconocer patrones ventajosos o ganadores en los juegos de suma–cero e información perfecta.
- Caracterizar qué definen estos patrones ventajosos.
- Cómo valorar esos patrones ventajosos, posiblemente a partir de una evaluación numérica.
- Cuáles son los elementos comunes que comparten estos patrones.
- La posibilidad de generalizar los patrones como una solución.

Para comprobar la hipótesis, se investigan los patrones primarios que definen las posiciones ventajosas o visiblemente ganadoras. El tipo de configuraciones que pueden presentarse así como los elementos generadores de las mismas. Se busca entonces establecer qué elementos las componen. Desde luego, se trabaja sobre juegos conocidos de suma–cero, como el NIM, el Gato y el ajedrez, por ejemplo, pero en última instancia se busca generalizar los *resultados* a cualquier juego de suma–cero e información perfecta.

Para ello, para el caso particular del ajedrez, se desarrolla un lenguaje de descripción de patrones, el cual permite definir sin ambigüedad, las configuraciones de las piezas en un tablero de ajedrez, generalizando las acciones de ataques y defensas que pueden realizar. Estos son los patrones propuestos como ventajosos o como aquéllos en donde uno de los jugadores tiene una ventaja tangible la cual podemos considerar incluso ganadora.

Se encuentra la existencia de muchos patrones ventajosos que en la literatura ajedrecística se encuentran catalogados (de manera muy informal) en la forma de ejercicios de táctica, donde el enunciado de las posiciones indica que a partir de la posición dada, uno de los rivales juega y gana (o empata). A partir de eso se desarrollan patrones[38] para las configuraciones más comunes en el ajedrez, [59] (capítulo 4). A partir de estos resultados, se trabaja

por lo tanto sobre un lenguaje de descripción para las posiciones de ajedrez, donde los elementos que conforman los patrones, es decir, las configuraciones ventajosos, puede describirse sin ambigüedad [60], (ver capítulo 6).

Finalmente se formaliza el lenguaje de descripción como un lenguaje algebraico. Un lenguaje de esta naturaleza, con una sintaxis y semántica precisa, se denomina un lenguaje formal (capítulo 7).

8.3. Contribuciones

El enfoque tradicional en el estudio de los juegos de suma–cero e información perfecta ha sido la la creación de árboles de movimientos y jugadas, y –a través de una función de evaluación– definir qué jugador tiene la ventaja, a partir de los valores que se entreguen en los nodos terminales, esto es, el algoritmo *Minimax* (Capítulo 2 – Algunas definiciones de teoría de juegos).

- Una primera contribución es el poder expresar los patrones de los juegos de suma–cero como patrones, los cuales se basan en gran medida en los trabajos del arquitecto Christopher Alexander [3], y se refieren a soluciones a problemas comunes, que ocurren una y otra vez en el contexto de la arquitectura y que se han pasado al diseño de software. Alexander define un patrón como **una regla de tres partes, la cual expresa una relación entre cierto contexto, un problema y una solución** (Capítulo 4) [3, 59].

- El análisis realizado en este trabajo muestra que la identificación de patrones ventajosos es una alternativa posible para hallar una heurística que muestre un comportamiento de buenas jugadas a realizar en los respectivos juegos de suma–cero. La contribución parte de evitar tener que analizar todo el árbol de variantes y valorarlo nodo por nodo y con diversos algoritmos para saber cuál es la mejor jugada. Nuestro enfoque puede orientar al jugador (o incluso a un programa de computadora), a hallar una solución eficiente a los problemas planteados en el juego (Capítulo 5 –Funciones de evaluación *versus* patrones).

- Para el caso del ajedrez, se define un lenguaje de descripción de patrones ventajosos, el cual es aplicado a diversas posiciones en donde se puede observar cómo los patrones existen y se tratan cada uno de ellos de forma similar (Capítulo 3 – Trabajo relacionado). El lenguaje

de patrones generaliza particularmente el problema de las configuraciones ventajosas y da una solución general al problema en lo que se refiere a tratar con heurísticas (Capítulo 7 – Descripción del lenguaje de patrones) [60].

- Una contribución más parte del formalizar este lenguaje de patrones para juegos de suma–cero e información perfecta en un álgebra de conjuntos, la cual se muestra como no-conmutativa y que puede representarse fácilmente en los lenguajes de programación, particularmente en aquellos que se denominan declarativos o funcionales (capítulo 7 – Hacia un álgebra de conjuntos para los patrones en ajedrez).

8.4. Trabajo futuro

A partir de la investigación realizada pueden plantearse nuevos temas a analizar. El primero se basa en presentar el lenguaje de patrones como un álgebra no conmutativa y ver hasta qué grado cumple con las características de la misma. El algebra no conmutativa estudia objetos algebraicos con una operación que no satisface la propiedad conmutativa, con especial énfasis en la Teoría de Anillos pero también con la vista en la aplicación en otras ramas de las matemáticas como por ejemplo la Teoría de Grupos, la Geometría o la Topología [7].

Otra vertiente para trabajo futuro es el estudio de la similitud de los patrones. ¿Cuándo un patrón es similar a otro? ¿Puede presentarse un grado de similitud –una métrica– que nos indique qué tan parecidos son? Este enfoque podría mostrar cuándo los patrones son similares y por qué.

Apéndice A

Definición formal de la notación ajedrecística

En muchas ramas de la ciencia, matemáticas, lingüística y computación, un lenguaje formal consiste en *palabras* cuyas *letras* se toman de un alfabeto definido y se construyen de acuerdo a un conjunto de reglas. El alfabeto de un lenguaje formal consiste en símbolos, letras o bloques que se concatenan en cadenas de caracteres (*strings*) que forman el lenguaje. Cada cadena concatenada de símbolos de este alfabeto se llama una *palabra*, y las palabras son entonces parte de un lenguaje particular formal, al que se le llama *palabras bien formadas* o *fórmulas bien construidas*. Así pues, un lenguaje formal se define con frecuencia por una gramática formal, como puede ser una *gramática regular* o una *gramática libre de contexto*, la cual consiste en sus reglas de formación. [50]

En los lenguajes de programación, la versión formalizada se refiere a subconjuntos del lenguaje natural en los que las palabras del lenguaje representan conceptos que están asociados con significados particulares, que es precisamente la *semántica*.

Un *alfabeto*, en el contexto de los lenguajes formales, puede ser cualquier conjunto de símbolos, aunque con frecuencia se usan las letras y números, que forman las *palabras* del lenguaje en cuestión. Por ejemplo, en los lenguajes de programación se usan el conjunto de caracteres ASCII o Unicode. Hay que decir que en la mayoría de las definiciones de los lenguajes formales, los alfabetos especificados tienen un número finito de elementos [50].

A.1. La anotación PGN

La anotación ajedrecística moderna, llamada "Portable Game Notation" - PGN, se define para poder registrar y reproducir las partidas de ajedrez. Fue diseñada por Steven Edwards en 1994[1].

La notación tiene dos acepciones: la notación larga (LAN - Large Algebraic Notation) y la notación corta (SAN - Short Algebraic Notation)[2][3].

La notación algebraica usa las letras $\{a, b, c, d, e, f, g, h\}$ para las columnas, siempre desde el punto de vista de las blancas, y de izquierda a derecha. Usa números $\{1, 2, 3, 4, 5, 6, 7, 8\}$. Definimos las piezas en el tablero como $\{P, K, Q, R, B, N\}$[4], P = peón, K = Rey, Q= Dama, R = Torre, B= Alfil y N = Caballo.

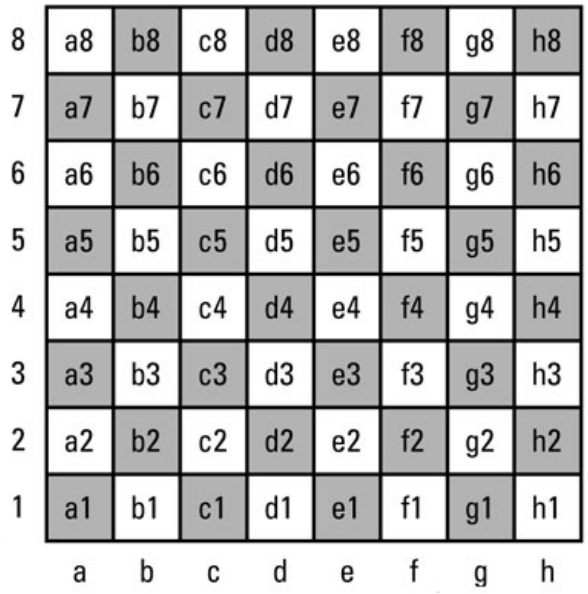

Figura A.1: El sistema algebraico para anotar partidas de ajedrez.

La notación larga (LAN), se expresa de la siguiente manera:

[1]La referencia completa está en https://www.thechessdrum.net/PGN_Reference.txt

[2]https://www.chessprogramming.org/Portable_Game_Notation

[3]https://www.linuxtopia.org/online_books/programming_books/python_programming/python_ch42.html

[4]la notación original de ajedrez está formalmente definida para las piezas en inglés

[**P**] *frmfr* [**n**]

P es el nombre de la pieza (se omite si se trata de peones), f es la columna (a-h), r es a fila (1-8), m es el movimiento (- ó x) y n es alguna de las siguientes anotaciones: +, #, !, !!, ?, ??. Las notas pueden incluir un símbolo "=" y la pieza de una letra cuando un peón llega a la octava fila.

La notación corta (SAN), omite la columna y fila inicial, a menos que sea necesario esto cuando exiata una posible ambigüedad. La sintaxis se define así:

[**P**] [**m**] [**d**] *fr* [**n**]

El movimiento m, solamente se especifica si es una captura (x). La d es, o una columna o una fila (preferentemente), o ambas, para elegir la pieza que se está moviendo cuando hay múltiples posibilidades.

En ambas notaciones, el enroque corto y largo se escribe *O-O* o *O-O-O*. De manera similar, el resultado del encuentro se anota como 1-0 (gana el blanco), 0-1 (gana el negro y 1/2-1/2 (empate). También se usa el símbolo "*****" cuando el juego no tiene aún resultado o es desconocido éste.

El estándar para las *notas* es "$" seguido de un número. Estos son los más usuales:

+ = JAQUE
# = JAQUE MATE
! = BUENA JUGADA
!! = JUGADA BRILLANTE
? = MALA JUGADA
?? = GRAVE ERROR

Apéndice B

Sistema de clasificación ELO

El sistema de clasificación ELO es un método matemático propuesto a partir de la estadística, para calcular la habilidad relativa entre los jugadores de ajedrez. Su inventor fue Arpad lo (1903 – 1992), físico estadounidense de origen húngaro. El método de Elo se adoptó en 1960 por la Federación estadounidense de ajedrez y para 1970 fue el método oficial de la FIDE – Federación Internacional de Ajedrez, por sus siglas en francés.

La puntuación o *rating* Elo se determina en términos de sus resultados contra otros jugadores. Todo se basa en asumir que la probabilidad de ganar una partida entre dos jugadores del mismo nivel es de, exactamente, el 50 por ciento. Elo define la puntuación esperada o expectativa, de acuerdo a la diferencia en puntos de rating. Una diferencia de 100 puntos de rating le otorga al jugador con más Elo una probabilidad de ganar del 70 por ciento.

La FIDE otorga títulos honoríficos cuando se llega a determinado rating (o Elo). Por ejemplo:

- 2300–2399 Maestro FIDE (MF)
- 2400–2499 Maestro Internacional (MI)
- 2500–2599 Gran Maestro (GM)

A partir de los 2600 puntos el único título que existe es el de Campeón del Mundo, quien es en general el jugador con más alto Elo en la lista oficial de la FIDE pero que además ha ganado el título enfrentando al Campeón mundial anterior. Magnus Carlsen, de Noruega, es el actual campeón del mundo con alrededor de 2855 puntos Elo [36].

Bibliografía

[1] Bruce Abramson and Richard E. Korf. A model of two-player evaluation functions. *Proceedings of the 6th National Conference on Artificial Intelligence*, 1987.

[2] C. Alexander. *Notes on the Synthesis of Form.* Harvard University Press, 1974.

[3] Christopher Alexander. *The Timeless Way of Building.* Oxford University Press, 1979.

[4] Paulo Brumaar Alexandre Linhares. ünderstanding our understanding of strategic scenarios: What role do chunks play?". *Cognitive Science*, 31:989–1007, 2007.

[5] Pablo Amster and Juan Pablo Pinasco. *Teoría de Juegos, una introducción matemática a la toma de decisiones.* Fondo de Cultura Económica, 2014.

[6] D.L.G. Anthony. *Patterns in Classroom Education. Pattern Languages of Program Design 2.* Addison Wesley, 1996.

[7] M. Atiyah. *Introducción al álgebra conmutativa.* Ed. Reverté, 1980.

[8] Yuri Averbakh. *Chess Endings: Essential Knowledge.* Pergamon Press, 1966.

[9] John Backus. Can programming be liberated from the von neumann style? a functional style and its algebra of programs.

[10] K. Beck, R. Crocker, J. Coplien, L. Dominic, G. Meszaros, E. Paulisch, and J. Vlissides. Industrial experience with design patterns. *Proceedings of ICSE*, 1996.

[11] Hans Berliner. *The System*. Gambit Books, 1999.

[12] Ken Binmore. *Fun and Games, A Text on Game Theory*. DC Heath, 1992.

[13] Ken Binmore. *Playing for Real: A text on game theory*. Oxford University Press, 2007.

[14] Mijail Botvinnik. *Computers, Chess and Long–Range Planning*. Springer–Verlag, 1970.

[15] CL Boulton. Nim, a game with a complete mathematical theory. *The Annals of Mathematics*, 1901.

[16] Charles L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3, no. 1/4:35–39, 1902.

[17] I. Bratko, P. Tancig, and S. Tancig. Detection of positional patterns in chess. *Advances in Computer Chess*, 4:113–126, 1986.

[18] Abramson Bruce. *The Expected Model of Two-Players Games*. Pitman, 1991.

[19] Frank Buschmann, Regine Meunier, Hans Rohnert, and Peter Sommerlad. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley and Sons, 1996.

[20] M.S. Campbell and H.J. Berliner. Using chunking to play chess pawn endgames. *Artificial Intelligence*, 23:97–120, 1984.

[21] W.G. Chase and H.A. Simon. Perception in chess. *Cognitive Psychology*, 4:55–81, 1973.

[22] P. Ciancarini and M. Gaspari. A knowledge based system and a development interface for the middle game in chess. *Advances in Computer Chess*, 5, 1989.

[23] Mike Clark. *Some Ideas for a Chess Compiler*. Springer Verlag, 1988.

[24] J.O. Coplien. *Advanced C++ Programming Styles and Idioms*. Addison Wesley, 1992.

[25] J.O. Coplien. *A Development Process Generative Pattern Language. Pattern Languages of Program Design*. Addison Wesley, 1995.

[26] G. Coste. The chess query language: Cql. *ICGA Journal*, 27:217–225, 2004.

[27] Stefano Crespi Reghizzi. *Formal Languages and Compilation*. Springer-Verlag, 2009.

[28] W. Cunningham. *The CHECKS Pattern Language of Information Integrity. Pattern Languages of Program Design*. Addison Wesley, 1995.

[29] Adriaan de Groot. *Thought and choice in chess*. Mouton, 1965.

[30] J.P. Marques de Sá. *Pattern Recognition - Concepts, Methods and Applications*. Springer-Verlag, 2001.

[31] D.L. DeBruler. *A Generative Pattern Language for Distributed Processing. Pattern Languages of Program Design*. Addison-Wesley, 1995.

[32] Jim E. Greer Dinesh Gadwal and Gordon I. McCall. Tutoring bishop-pawn endgames: An experiment in using knowledge-based chess as a domain for intelligent tutoring. *Applied Intelligence 3*, 3, 1996.

[33] Chrilly Donninger. Che: A graphical language for expressing chess knowledge. *ICCA Journal*, 12, 1996.

[34] Mark Dvoretsky. *Dvoretsky's Endgame Manual*. Russell Enterprises, 2006.

[35] Susan Eisenbach. *FUNCTIONAL PROGRAMMING: Languages, Tools and Architectures*.

[36] Arpad Elo. *The Rating of Chessplayers*. Bastford, 2000.

[37] John Emms. *The Survival Guide to Rook Endings*. Gambit Publications, 2008.

[38] C. Alexander et al. *A Pattern Language*. Oxford University Press, 1977.

[39] Max Euwe. Mathematics – set theoretic consideratrions on the game of chess. *New Mathematics and Natural Computation*, 12(1), 2016.

[40] Peter W. Frey. *Chess Skill in Man and Machine*. Springer-Verlag, 1977.

[41] Peter W. Frey. Fuzzy production rules in chess. *ICCA*, 9, 4, 1986.

[42] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[43] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Addison Wesley, 1995.

[44] Debasis Ganguly, Johannes Leveling, and Gareth J.F. Jones. Retrieval of similar chess positions. *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 687–696, 2014.

[45] Michael George and Jonathan Schaeffer. Chunking for experience. *ICCA Journal*, 13:123–132, 1990.

[46] R. Gibbons. *A Primer in Game Theory*. Harvester Wheatsheaf, 1992.

[47] F. Gobet and M. Bibalic. They do what they are told to do: The infuence of instruction on (chess) expert perception - commentary on linhares and brum.

[48] F. Gobet and H. A. r Simon. Templates in chess memory: A mechanism for recalling several boards. *Cognitive Psychology*, 31, 1996.

[49] F. Gobet and H. A. r Simon. Expert chess memory: Revisiting the chunking hypothesis. *Memory*, 6(3), 1998.

[50] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[51] Christian Hesse. *The Joy of Chess*. New in Chess, 2011.

[52] Alese B.K. Ogundele O.S. Ibidunmoye, E.O. Modeling attacker-defender interaction as a zero–sum stochastic game.

[53] José Luis Jimeno y Emilio Cerdá Joaquín Pérez. *Teoría de juegos*. Pearson, Prentice Hall, 2004.

[54] Michael Kearns. A tutorial of computational game theory. *Class notes*, 2002.

[55] Donald E. Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7-12, 1964.

[56] Bent Larsen and Steffen Zeuthen. *ZOOM 001, Zero Hour for Operative Opening Models*. Dansk Skakforlag, 1979.

[57] A. Linhares and P. Brum. How can experts see the invisible? reply to bilalic and gobet. *Cognitive Science*, 33:5, 2009.

[58] Alexandre Linhares. The emergence of choice:decision-making and strategic thinking through analogies. *Information Sciences*, 259:2, 2008.

[59] López–Michelone M. and Ortega–Arjona J.L. Patterns for the game of chess. *Proc. of Latin American Conf. on Pattern Lang. of Prog.*, 11, 2016.

[60] López–Michelone M. and Ortega–Arjona J.L. A description language for chess. *ICGA Journal*, 42(1), 2020.

[61] J. C. C. McKinsey. *Introduction to the Theory of Games*. Dover, 2003.

[62] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[63] John Nunn. *Secrets of Rook Endings*. Batsford, 1993.

[64] Michael L. O'Lealy. *A First Course in Mathematical Logic and Set Theory*. Wiley, 2016.

[65] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT, 1994.

[66] D.L. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering, SE-2*, 1, 9, 1976.

[67] E.D. Posnyanskaya and O.K. Tikhomirov. On the function of eye movements. *Soviet Psychology*, 1:25–30, 1969.

[68] C.J.S. Purdy. *C.J.S. Purdy on the Endgame*. Thinker's Press, 2003.

[69] T. E. S. Raghavan. *Handbook of Game Theory*, volume 2. Elsevier, 1994.

[70] Eric Rasmusen. *Games and Information: An Introduction to Game Theory*. Wiley, 2006.

[71] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[72] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[73] Francisco Sánchez Sánchez. *Introducción a la matemática de los juegos*. Siglo XXI Editores, 1993.

[74] Jonathan Schaeffer, Neil Burch, Yngvi Bjornsson, Akihiro Kishioto, Martin Muller, Rob Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 2007.

[75] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41, No. 314, 1950.

[76] Andrew Soltis. *100 Chess Master Trade Secrets: From Sacrifices to Endgames*. Batsford, 2014.

[77] Siegbert Tarrasch. *The Game of Chess*. Dover, 1988.

[78] E Theodore L. Turocy and Bernhard von Stengel. Game theory. *CDAM Research Report LSE-CDAM-2001-09*, 2001.

[79] P. L. Villagra and F. Jakel. Categorization and abstract similarity in chess. *Proceedings of the COGSCI 2013*, pages 2860–2866, 2013.

[80] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[81] Murray Capbell y Hans Berliner. A chess program that chunks. *AAAI Procedings 1983*, 1983.

[82] C.C. Yang. Content-based image retrieval: A comparison between query by example and image browsing map approaches. *J. Information Science*, 30 (3):254–267, 2004.

[83] Paul Zhen Ni, Shuva. A multistage game in smart grid security: A reinforcement learning solution.

[84] A.L. Zobrist and Jr. F.R. Carlson. An advice-taking chess computer. *Scientific American*, 228, no. 6:92–105, 1973.