CM

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS

del

Instituto Politécnico Nacional

Departamento de Ingeniería Eléctrica

Sección de Computación

Deducción basada en consejos y aprendizaje automático.

Tesis que presenta el Licenciado en Computación

Mauricio Javier Osorio Galindo

para obtener el grado de :

Maestro en Ciencias

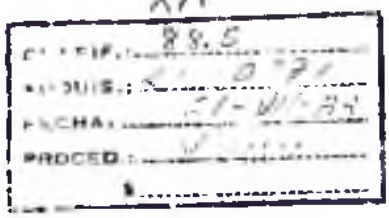
en la especialidad de Ingeniería Eléctrica

Trabajo dirido por los doctores

Zdenek Zdrahal y Guillermo Benito Morales Luna

Mexico D.F. Abril de 1988

XM



I N D I C E .

2.5 - Conclusiones generales sobre AL1

- Ventajas de AL1
- Niveles de abstracción en AL1
- Prolog y AL1
- Posible reforma a la definición de árbol forzado

2.6 Implementaciones

- Micro-Prolog
- Pascal

3. - AL3 (lenguaje de consejos 3)

3.1 - Definición de AL3

- Vista general de AL3

3.1.1 - Antecedentes a AL3

- AL1.5
- AL2
- Restricciones compartidas por AL1,AL1.5,AL2

3.1.2 - Elementos en AL3

- Método, Lema, Consejos, Arbol forzado, satisfactibilidad
- Plan, Exito de un plan, Refutación
- Utilidad de estos conceptos en ajedrez

3.1.3 - AL3 como sistema experto

- Módulo de control
- Base de conocimientos

3.1.4 - Metaconocimiento sobre planes en AL3

3.2 - Descripción del sistema y su uso

3.2.1 - Arquitectura de AL3

- Nivel general de operació de AL3.
- Traductor de código fuente a código intermedio
- Máquina inferencial

3.2.2 - Fundamentos teóricos del sistema

- Manejo de las funciones lógicas
- Actualización de los objetivos
- Operaciones que simplifican los objetivos

3.2.3 - Ejemplo en AL3

- Final de peón y rey contra rey

3.2.4 - Evaluación y eficiencia de AL3

3.2.5 - Conclusiones de AL3 y posibildades de desarrollo

- AL3 como un resolvedor general de problemas
- AL3 orientado al ajedrez

4. - Aprendizaje

- Introducción
- Algunos modelos de aprendizaje

- Conclusiones generales

Apéndices

- Mate de torres en Convert sugiere una aportación más general
- Notación algebraica y descriptiva para ajedrez
- Algunos ejemplos del ajedrez muestran su complejidad
- Programación de uso de AI3
- Listados AI3

Bibliografía.

Prefacio

El trabajo que a continuación se expone tiene por objetivo poner a prueba mecanismos de "razonamiento" y "planeación" en una computadora. Se utiliza un lenguaje apropiado para representar el conocimiento en sus diversas formas, tales como consejos, planes, métodos,hipótesis, etc.

"Lenguaje de consejos" (Advice Languaje) es el nombre de este leguaje y en términos generales es una herramienta para la representación de heurísticas y estrategias para la solución de problemas (problem-solving) .

El trabajo de tesis consistió en :

- Estudiar los métodos tradicionales de solución automática de problemas. Principalmente los utilizados en juegos con estrategia.

- Estudiar, implementar y utilizar AL1 que es la primera versión de los lenguajes de consejos. Más concretamente :

  - Se implementó AL1 en Micro-Prolog.
  - Se desarrollaron dos tablas de consejos, una para dar mate con torre y rey vs rey y otra para dar mate con 2 alfiles y rey vs rey.
  - Se probaron estas tablas con el sistema AL1 en Micro Prolog.
  - Se implementó un traductor de AL1 a Pascal.
  - Se probó con el ejemplo de mate con alfiles y se comparó con la versión en Micro_Prolog.
  - Se discute una posible reforma (muy ligera) al concepto de árbol forzado . Elemento central en los lenguajes de consejos.

- Estudiar, implementar y utilizar AL2 que es la versión actual de los lenguajes de consejos. El capítulo 3 corresponde a esta parte. La implementación fue hecha en Micro-Prolog y se definió una base de conocimientos de finales de rey y peón vs rey.

- Estudiar algunos métodos de aprendizaje (en particular orientados a juegos) y se discute la posibilidad de utilizarlos en los lenguajes de consejos.

Dominio de Experimentación

De acuerdo con la estrategia de Michie para la investigación en la Ingeniería del Conocimiento, el ajedrez se usa como un dominio de prueba. Se pueden mencionar algunas razones para esta elección.

- Las reglas son bien definidas y fáciles de aplicar ,lo que nos permite abstraer estos aspectos que inicialmente son secundarios.

- Una gran cantidad de conocimiento de ajedrez ha sido registrado en la literatura, el cual puede ser usado y a su vez confrontado.

- Puede hallarse una gran variedad de problemas que son diferentes en términos de sus estrategias para su solución.

- Las capacidades mentales que tiene un experto en ajedrez para resolver un problema de ajedrez no parecen exclusivas para esta área sino que son esenciales para resolver una amplia variedad de problemas, ya sea en medicina, ingeniería o en cualquier otra disciplina científica [Frey y Atkin 1978].

Por lo anterior puede decirse que "el ajedrez contiene cualidades interesantes para utilizarlo como laboratorio en la solución de problemas " .

Base o Núcleo de AL:

Se intenta representar el conocimiento en base al sentido común mediante 'consejos', los cuales sugieren qué meta debe ser alcanzada inmediatamente y una guía para lograrlo.

Por lo tanto se trata de descomponer la solución de un problema en una secuencia de metas intermedias y para alcanzar a cada una de ellas se cuentan con consejos (sugerencias) para hacerlo. Más concretamente un consejo consiste de :

Better Goal	Metas intermedias
Holding Goal	Condiciones que deben mantenerse
Move constraints	Subconjunto ordenado de operadores

1. Cómo juegan Ajedrez las Computadoras .

1.1. Introducción

Dentro de la ideas para la programación del juego de ajedrez se encuentra que se han utilizado fundamentalmente dos enfoques los cuales son:

A) Fuerza Bruta (búsquedas exhaustivas)

B) Ingeniería del Conocimiento

El primero en surgir y hasta la fecha es el utilizado en los programas comerciales fue el enfoque (A), mientras que el segundo es más reciente y últimamente más estudiado debido a la proliferación de los sistemas expertos.

1.2. Uso de Fuerza Bruta en el Ajedrez

Al observar una posición de Ajedrez cualquier ajedrecista puede opinar si es buena o mala para alguno de los bandos sin tener que recurrir a una inspección de variantes. El criterio de decisión dependerá del nivel ajedrecístico, así en un nivel elemental se recurre al balance material, en cambio los jugadores más experimentados pueden recurrir a más parámetros para su juicio, tales como estructura de peones, control del centro movilidad, seguridad del rey, etc.

Asimismo se espera que las computadoras puedan evaluar posiciones en forma estática. Para ésto se utilizan funciones que asocian números a las posiciones de ajedrez para alguno de los bandos. Se espera que los números revelen qué posiciones son mejores que otras.

A continuación se muestra un ejemplo muy sencillo, tal vez rústico, de una función de evaluación propuesta por Shannon [Newborn 75].

Los parámetros son material, estructura de peones y movilidad.

MATERIAL. La mayoría de las funciones de evaluación utilizadas hasta la fecha dan mucha importancia a este aspecto.

Las piezas se valoran como sigue:

Dama=9, Torre=5, Alfil=3, Caballo=3, Peón=1, Rey=200

ESTRUCTURA de peones.

Se desea evitar peones doblados, sueltos y retrasados. Así que por cada uno de estos casos que se presenten,Shannon sugirió 1/2 punto de castigo.

MOVILIDAD.

Agréguese .1 punto por cada movimiento legal disponible. El valor S de una posición Pos para el blanco, denotado por S(Pos) queda dado por la siguiente función de evaluación:

$$S(Pos)=200(R-R')+9(D-D')+5(T-T')+3(A-A'+C-C')+$$
$$(P-P')+.5(PD-PD'+PS-PS'+PR-PR')+.1(M-M')$$

donde R,D,T,A,C,P son el número de reyes,damas torres,alfiles,caballos y peones blancos respectivamente.

PD,PS,PR representan el número de peones doblados,sueltos y retrasados del blanco.

M es el número de movimientos legales del blanco.

Las variables primas representan variables similares para el negro. Los valores positivos indican ventaja del blanco, mientras que negativos indican ventaja del negro.

Para lograr resultados satisfactorios es necesario definir adecuadamente una función de evaluación, por lo que se invita a todos los interesados en el tema a que traten de crear la suya propia.

Tomando como base una función de evaluación podemos definir una estrategia de un movimiento hacia adelante de la siguiente manera: Para una posición P supóngase que se pueden realizar r movimientos legales denotados por M1,M2,...,Mr .Ahora constrúyanse las r posibles posiciones como si se realizara cada movimiento. LLamemos a estas posiciones M1P,M2P,...,MrP.Aplicando la función de evaluación a cada una de es as po iciones obtenemos S(M1P),S(M2P),...,S(MrP). Finalmente, si es el turno del blanco elíjase el movimiento que conduzca a la posición con el máximo valor y si el turno es del negro se escoge el movimiento que conduzca a la posición con el mínimo alor.

El árbol en la figura 1 ilustra esta estrategia para una posición hipotética P teniendo el blanco 4 movimientos legales M1,M2,M3 y M4. El nodo a la izquierda representa la posición P y es llamado la raíz del árbol. Los 4 movimientos son representados por ramas que conectan al nodo de la izquierda con los nodos terminales de la derecha. Los nodos terminales representan posiciones M1P,M2P,M3P y M4P. La puntuación de cada nodo terminal se muestra al lado de él.

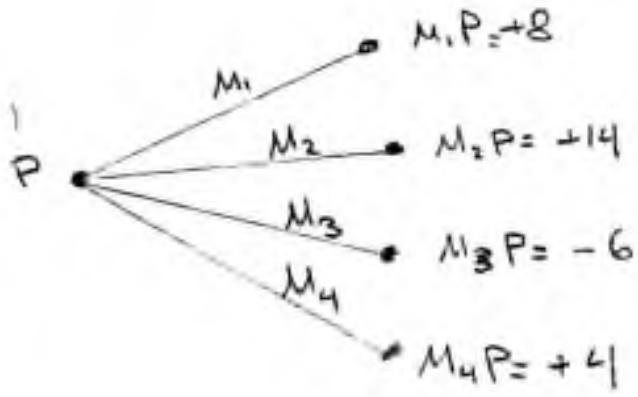


Figura 1

De lo anterior, el mejor movimiento del blanco es M2 pues nos conduce a la posición con puntuación de 14 que es la más alta. De esta manera decimos que a la posición P se le puede asignar una puntuación +14 por retroceso. En general, el blanco elige el movimiento de tal manera que la puntuación de retroceso para la posición P sea máxima. Para el negro se escoge la mínima.

A continuación definiremos una estrategia de 2 movimientos hacia adelante de la siguiente manera:

Obténganse todos los movimientos legales del blanco (suponiendo que es su turno) para la posición P. Llamémoslos M1,M2,...,Mr. Contrúyanse a continuación las posiciones noterminales M1P,M2P,....,.MrP y apliquese la estrategia de un movimiento hacia adelante para el negro en cada una de estas posiciones. Obtendremos así:

1.- La mejor respuesta del negro según el movimiento del blanco.

2.- La puntuación de retroceso para cada una de las posiciones M1P,M2P,...,MrP

Finalmente basados en las puntuaciones de retroceso para las posiciones noterminales mencionadas en el punto 2 podemos aplicar

la estrategia de un movimiento hacia adelante para el blanco en la posición P.Recurramos a un ejemplo muy sencillo para aclarar posibles dudas.Supóngase la siguiente posición (figura 2):

Blancas R6TD,P7CD y Negras R2AD

y una función de evaluación basada en el balance material definida como :

S(P)= número de peones blancos - número de peones negros

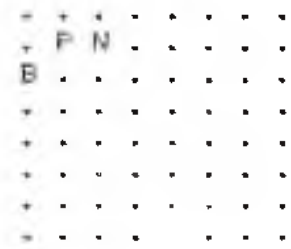


Figura 2

Apliquemc el procedimiento por pasos:

1.- Obtenemos las continuaciones legales del blanco, es decir

M1=R7T y M2=R5T (descartemos R5C y P8C , para simplificar el problema)

2.- Construimos las 2 posiciones noterminales

M1P Blancas R7T,P7C y Negras R2AD

M2P = Blancas R5T,P7C y Negras R2AD

3.- Aplicamos la estrategia de un movimiento hacia adelante para el negro en las 2 posiciones.

Para M1P obtenemos que la mínima puntuación que podemos lograr es 1. De hecho esta puntuación se logra con cualquier continuación del negro pues no puede modificar el balance material Por tanto la puntuación de retroceso para M1P es 1.

Para M2P obtenemos que la mínima puntuación que podemos

lograr es 0 pues es posible capturar el peón. Por tanto la puntuación de retroceso para M2P es 0.

4.- Aplicando ahora la estrategia de un movimiento hacia adelante para el blanco para la posición P siendo que M1P=1 (abreviando la notación más formal S(M1P)=1) y M2P=0 obtenemos que la máxima puntuación que puede lograrse es 1 por medio de M1P.

Por tanto la puntuación de retroceso para P es 1.

El movimiento que se elige entonces es M1 o sea R7T, que es precisamente lo que todos esperábamos. La figura 3 muestra un árbol para este caso.

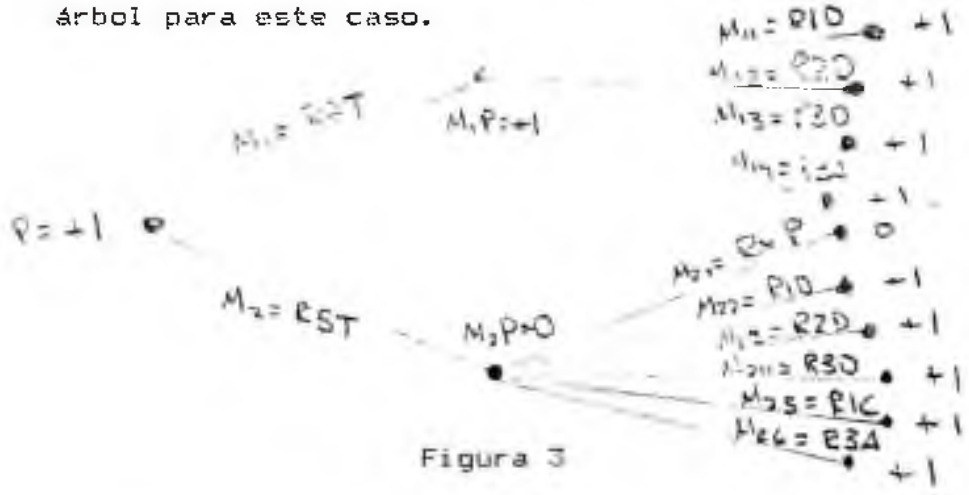


Figura 3

resto de movimientos son :

M22= R1D, M23= R2D, M24=R3D, M25= R1C, M26= R3A

Así como se definieron estrategias para 1 o 2 movimientos hacia adelante se puede definir una estrategia para n movimientos hacia adelante. Esto puede hacerse mediante la siguiente definición recursiva :

V(X,n) = S(X) si n=0 o X es una posición final, es decir

Jaque Mate o Ahogado

siendo S(X) la evaluación estática

$$V(X,n) = \max \{ -V(Hi,n-1) \} \text{ para } 1 <= i <= d \quad \text{si } n > 0$$

siendo Hi,H2,..., Hd los hijos de X

Nótese que se logra el efecto de evaluar alternadamente el máximo y mínimo mediante el cálculo de max con argumentos negados.

Este método se le conoce como método minimax, haciendo algunas mejoras de tal manera que no tengamos que recorrer todo el árbol y el procedimiento resultante se conoce como alfa-beta [Horowit: 76]. Observemos en la figura 4 el comportamiento de este algoritmo. La computadora iniciará la búsqueda minimax examinando la posición P y efectuando los movimientos M1 y M2 . Entonces generará la posición M1P y a continuación los movimientos M11P, M12P. Dado esto, obtendrá las puntuaciones para M11P y M12P , concluyendo que si realiza el movimiento M1, su oponente jugará M12. Lo que conduciría a una puntuación de retroceso de +7 para el movimiento M1. Ahora la computadora analizará la posición M2P con los movimientos M21 , M22 , M23 y M24 . Obtendrá así la puntuación para M21P y después la puntuación para M22P . En este momento, habiendo hallado que la puntuación para M22P es menor que +7 (puntuación de M1P) la computadora concluye que el movimiento M2 no es tan bueno como M1. El movimiento M22 se entiende como una refutación al movimiento M2. De esta manera ya no hay caso en examinar los movimientos M23 y M24. Sus puntuaciones son irrelevantes . De cualquier forma la puntuación de retroceso para P es +7. Decimos que el movimiento M22 causa un corte en la búsqueda en la posición M2P.

Figura 4

El procedimiento alfa-beta en algún pseudo-código de alto nivel puede ser a grandes rasgos el siguiente :

```
procedimiento AB(X,N,C)
/*  X es la posición , N el límite de movimientos,
    C es la cota que permite los cortes , terminal(X) es
    una función que verifica posiciones finales como Jaque
    Mate y Ahogado , S(X) es la evaluación estática  /*
SI terminal(X) o N=0 entonces regresa S(X)
TEMP= - 10e10    /*  - infinito  /*
PARA TODO Hi , tal que Hi es hijo de X efectúa
  { TEMP = max { TEMP, -AB(Hi,N-1,-TEMP) }
    Si TEMP >= C entonces regresa TEMP
  }
regresa TEMP
fin
```

La llamada inicial deberá ser AB(X,N,infinito) .

1.3. Cómo Piensan los Grandes Maestros.

Actualmente se ha demostrado que la creencia popular de que los grandes maestros calculan muchas jugadas por anticipado es inválido. No es la intención excluir la existencia de dichos casos: más bien se refiere a la generalidad.

A finales del siglo XIX Binet se sorprendió de que los maestos no tienen una imagen viva del tablero cuando juegan sin ver . En lugar de esto, parece ser que recuerdan las posiciones en forma abstracta , como por ejemplo , recordando relaciones entre las piezas. Posteriormente (entre los 1930s y 40s) De Groot trabajó con un grupo de jugadores de ajedrez de alto nivel (incluyendo grandes maestos) pidiéndoles que expresaran sus pensamientos mientras seleccionaban su movimiento en una posición complicada. Sus estudios revelaron que los grandes maestros operan en forma muy similar a los jugadores débiles. Ambos analizan con una anticipación similar de número de jugadas, ven aproximadamente el mismo número de posiciones diferentes, calculan combinaciones con profundidad similar y en general desde el punto de vista de capacidades mentales se comportan análogamente. La diferencia es que los grandes maestros invariablemente seleccionaban los mejores movimientos. Por tanto, se intuye que el proceso de análisis en si mismo no es un factor decisivo para determinar la fuerza del ajedrecista.

Trabajos más recientes, como los realizados por Chase y Simon en la universidad de Carnegie-Mellon muestran que los ajedrecistas fuertes tienen mejor memoria que la de de los novatos en ciertas posiciones de ajedrez, cualidad que desaparece cuando las piezas

se colocan en forma aleatoria.

Simon y Gilmartin proponen que los jugadores fuertes aprenden a reconocer un gran número de combinaciones entre piezas percibiendo patrones entre éstas.

Así, los grandes maestros, en lugar de recordar por ejemplo 5 o 6 posiciones ,como lo haría un jugador novato, recuerdan 5 o 6 patrones, que se traducen en muchas posiciones concretas del tipo que recordaría un jugador novato.

Hay otro punto muy interesante que han revelado estos trabajos con respecto a la elección del movimiento que realizan los grandes maestros y es que en pocos segundos obtienen un conjunto reducido de posibles movimientos en los que usualmente se halla el que después escogerán, luego de un análisis más agudo.

De manera que la habilidad humana para jugar ajedrez esta basada en dos capacidades altamente refinadas:

reconocimiento de patrones y rápido acceso a la información.

Capacidades que por otro lado no se consideran exclusivas para el ajedrez sino para cualquier rama del conocimiento.

1.4. Enfoque basado en la Ingeniería del Conocimiento

1.4.1. En que consiste la ingeniería del conocimiento

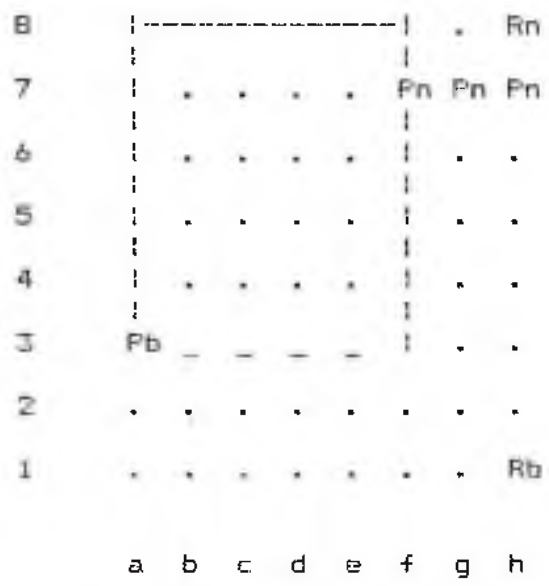
Se dice que la Ingeniería del Conocimiento es la que se refiere a las técnicas utilizadas para la representación, manipulación, adquisición de conocimiento apropiadas para la construcción y explicación de líneas de razonamiento sobre aplicaciones 'difíciles' que requieren conocimiento 'experto' para su solución [Hayes-Roth y Waterman 83].

La aplicación es el juego de ajedrez, para nuestro caso , donde parece muy relevante una buena manipulación del conocimiento para lograr resultados interesantes.

1.4.2. Importancia del conocimiento en ajedrez

Mostraremos lo valioso del conocimiento en el Ajedrez con un sencillo ejemplo mostrado en la figura 5.

Supongamos que en la siguiente posición juegan las blancas:



donde Rb = Rey blanco , Pb = Peón blanco

Rn = Rey negro , Pn = Peón negro

F i g u r a 5

Mediante el uso de la regla del cuadrado del peón, el blanco gana coronando su peón aunque el bando negro tenga ventaja material (2 peones). Esta regla establece que para poder frenar un peón libre, el rey defensor debe estar dentro del cuadrado (remarcado en la figura) cuando es el turno de movimiento del bando que posee dicho peón.

Utilizar la regla del cuadrado para la posición anterior es equivalente a lo que se descubriría en una búsqueda de 10 movimientos adelante (5 por cada bando) para encontrar que el peón blanco corona. Esto motiva la siguiente definición que sirve

para catalogar o valorar un cierto conocimiento en ajedrez.

Se define la capacidad de proyección de un "trozo de conocimiento" como el número promedio de movimientos que deben efectuarse antes de que éste conduzca a la meta señalada por el trozo de conocimiento dado [Berliner 84]. Una meta típica es la ganancia de un peón.

Considerar esta meta es debido a que 2 peones de ventaja se consideran en la mayoría de los casos, decisivos para ganar una partida (a un nivel de maestros) y la ventaja de un solo peón es decisiva en aproximadamente el 50% de los casos.

En términos ajedrecísticos se puede dividir la ventaja en dos grandes ramas : táctica y estratégica. Se considera que la capacidad de proyección táctica se encuentra entre 3 y 19 movimientos , mientras que la capacidad de proyección estratégica oscila entre 30 y 80 movimientos.

Regresando nuevamente al ejemplo de la figura 5 , si bien es cierto que la regla del cuadrado podría incluirse en una evaluación estática por medio del método alfa-beta , existirán inconveniencias como las siguientes. Supóngase que N (profundidad de análisis) es un número no pequeño. Entonces se calcularán muchas posiciones y movimientos en forma innecesaria, pues es inmediato saber que las blancas ganan. N tendría que ser pequeño para aprovechar el conocimiento y evitar búsquedas innecesarias (de hecho N sería 0 para utilizar únicamente el conocimiento sin recurrir a búsqueda alguna) pero en este caso que tal si la posición fuese otra donde el conocimiento es incompleto y se requiera hacer búsquedas ? Para adaptar el método alfa-beta N

tendría que ser función de la posición . Este simple hecho incrementa mucho la complejidad del problema. Por otro lado, valdrá la pena incluir en la función de evaluación la regla del cuadrado (tan particular) y tener que calcularla durante toda la partida en todas posiciones terminales (miles) ? Cómo incluirla como parte de una función número ica ? que peso le damos a este parámetro ? Se está complicando mucho un problema que para un ajedrecista novato es simple de resolver . Lo que se intenta subrayar es que el conocimiento no se representa ni se maneja muy convenientemente por medio de funciones de evaluación y búsquedas alfa-beta. La alternativa es pues un sistema que represente y maneje convenientemente el conocimiento. Pero cómo valorar o decidir si un sistema cumple con esta cuestión ? Parece que si el sistema es capaz de manejar `planes` en forma satisfactoria entonces cumple en buena medida con su misión.

1.4.3. Uso de Planes

El manejo de los planes en el ajedrez es fundamental para tener un dominio del juego.

Los planes ayudan a reducir el costo del procesamiento del conocimiento ya que inmediatamente centran su atención en la parte crítica de la nueva posición creada durante la búsqueda, evitando así un completo reanálisis de la posición generada. Teniendo planes podemos realizar mucha poda en el árbol guiando así la búsqueda .

Alexander Kotov [Kotov 83] lo define como :

> Un conjunto de operaciones estratégicas sucesivas, realizables según las ideas que sugieren las exigencias de la situación creada en el tablero.

Es tan importante el plan pare él, que hace los siguientes comentarios.

En [Kotov 71] scribe :

> . . . es mejor seguir un plan consistentemente, aún cuando éste no sea el mejor, que jugar adoleciendo de plan al . no La peor cosa es vagar sin propósito fijo .

Y en [Kotov 83] escribe:

> En este aspecto es muy característico el juego de Nimzowitsch, cuyos planes son a menudo paradójicos; sin embargo, tiene éxito porque los lleva a término con una tenacidad digna de envidia y sabe al propio tiempo eludir todo obstáculo táctico.

Nimzowitch ,por cierto, fue un gran ajedrecista y a él se debe gran parte de la teoría moderna del ajedrez.

A continuación se presenta un ejemplo de Plan en ajedrez :

Supóngase que se tiene que dar mate con torre y rey vs rey . Supongamos también que las blancas conducen a la torre y al rey.Los libros de ajedrez presentan tal plan consistiendo de las siguientes dos etapas:

1.- El rey y la torre perseguirán al rey de las negras hasta una columna o fila extrema.

2.- Por medio de jugadas forzadas, pues aquí no se puede ganar sin ellas, un rey se situará enfrente del otro, y la torre dará mate colocándose en forma perpendicular a la columna o fila donde se hallan los reyes.

Este plan de dos metas (una para cad· etapa ha simplificado el problema original pero aún puede resultar difícil saber cómo llevarlo a cabo. Por otro lado hay ciertos aspectos implícitos dentro del plan que conviene subrayar, algunos de éstos son :

Hay que cuidar no perder la torre, pues sin ella será imposible dar mate y se debe evitar caer en posiciones ahogadas pues en ajedrez sería empate (tablas en el argot del ajedrez).

Niblet se encuentra con esta dificultad cuando intenta definir un plan para vencer con torre y rey vs rey usando un libro de ajedrez. El señala :

> La representación del problema en el libro es lúcida para el jugador humano. Para un programa que juegue ajedrez es deficiente en muchos aspectos.
>
> Se omite información necesaria, usualmente la que es muy obvia.

Debe entonces ser más precisa la manera de enunciar planes para manejarse en términos computacionales. Creo también que muchos aficionados al ajedrez podrían verse beneficiados con definiciones más precisas sobre las estrategias en ajedrez.

Otra observación sobre el ejemplo del plan mencionado es que debemos de tener un mecanismo para detectar cuando estamos en la fase 1 o 2 para poder tomar la alternativa apropiada. Sería también conveniente definir condiciones que se deben mantener durante la fase 2 para no regresar a la fase 1. Hagamos algunos cambios al enunciado del plan que incluyan las últimas observaciones previas.

Nota : Usemos la palabra orilla para entender columna o fila extrema.

Defínase antes el siguiente patrón de posiciones :

a) el rey negro se halla en alguna orilla

b) los reyes se encuentran enfrente uno del otro, con una sóla casilla libre entre ellos y

c) que la torre se halle en la fila o columna perpendicular a los reyes que contiene la casilla que separa a éstos con cuando menos dos casillas libres entre la torre y esta casilla mencionada.

Llamemos a este patrón patrón_para_mate.

El plan (ahora de tres fases) puede entonces enunciarse como:

1. Si el rey negro no se halla en alguna orilla entonces

   Evitando perder la torre

   Búsquese llevar al rey negro hacia alguna orilla y que la torre se halle en la columna o fila contigua a la orilla donde se halla el rey negro

2.- Si el rey negro se halla en alguna orilla pero no tenemos un patrón_para_mate

   evitando perder la torre y

   evitando ahogar al rey negro y

   manteniendo la torre en la columna o fila contigua a la orilla donde se halla el rey negro para evitar que éste escape

   Búsquese que los reyes se hallen a una distancia de un movimiento de caballo

3.- Si la posición es un patrón_para_mate entonces

   Manteniendo la torre en la columna o fila contigua a la orilla donde se halla el rey negro para evitar que éste escape

   Búsquese mate con un sólo movimiento de torre

Obsérvese que cada fase del plan puede verse como un subplan con el siguiente esquema:

Si patrón_posición entonces

mantengase ciertas condiciones_necesarias

buśquese meta

El hecho de que en 3) no halla condiciones_necesarias no afecta el esquema, pues podrían definirse como nulas. Lo que sí es digno de remarcar es que precisamente en 3) la meta se definió con algunas peculiaridades que a continuación se resaltan :

a) Se indicó un cierto número de movimientos a realizar (uno).

b) Se indicó un subconjunto de las movimientos legales (mover tan solo la torre)

Considerando que un plan es un "trozo de conocimiento" se observa que a) coincide con la capacidad de proyección definida previamente. Esto da la pauta para sugerir una posible extensión del esquema de plan como :

Si patrón_posición entonces

{ manténganse ciertas condiciones necesarias

utilícese un cierto subconjunto de movimientos legales

proyección del plan

búsquese meta }

Recordando que el plan sugerido originalmente no indicaba cuando llevarse a cabo sino tan solo cómo realizarlo ,puede ser mejor dejar lo que está entre llaves como plan y a patrón_posición como criterio para seleccionar el plan. Todo junto (criterio y plan) corresponde al concepto computacional de 'regla' de un sistema de

producciones. Los sistemas de producciones son una forma conocida de representación del conocimiento. Herbert A. Simon escribe:

> Los sistemas de producción son un tipo particular de formalismo de programación que se considera particularmente apropiado para representar estrategias humanas.

1.5 - Conclusiones.

Tenemos que, el uso de planes por medio de reglas para manejar el conocimiento en ajedrez , parece adecuado para el manejo de la tática y la estrategia.

Manejar reglas en base a patrones parece ser más cercano al proceder de los maestros del ajedrez que utilizar funciones de evaluación sobre búsquedas alfa-beta . Concuerda esta observación con Michie , quien indica que las estrategias cognoscitivas de un humano experto en dominios muy complejos se basa :

> ... no en cálculos muy elaborados, sino en la memoria y el uso de grandes catálogos incrementales de reglas basadas en patrones.

Sin embargo , se observa que para el ajedrez el número de reglas que se requieren crecen relativamente poco, cuando se incrementa el dominio de complejidad [Chapa 84] .

Notemos por último que el esquema del plan sugerido parece interesante pero no necesariamente acabado o único.

2 AL1.

2.1 Definición de AL1.

Dentro los lenguajes orientados a la representación y manejo de planes se encuentra que AL1 se considera como una de las proposiciones más interesantes. Los elementos de AL1 serán descritos de lo general a lo particular.

Una tabla de consejos es el elemento más general. Será usada para resolver un problema especializado. En el Ajedrez, por ejemplo, se puede tener una tabla de consejos que es usada para resolver una posición de final de peón, otro para resolver una posición de de final de damas, etc tera. Una serie de reglas es lo que forma una tabla de consejos y cada regla es de la forma :

Si precondición entonces lista-de-consejos

Si más de una precondición es satisfecha entonces simplemente se escoge la primera regla entre éstas.

Continuaremos la explicación de los elementos con el siguiente problema de ajedrez: Blancas, las cuales tienen una torre y un rey deben dar jaque mate a las negras, quienes sólo tienen el rey.

Una tabla de consejos en este caso puede consistir
de las dos reglas siguientes.

Si el rey negro no está en el límite del tablero entonces intentar ponerlo en el límite del tablero

Si el rey negro está en el límite del tablero entonces intentar dar jaque mate

Para ejecutar la acción "intentar ponerlo en el límite del tablero" puede ser necesario intentar diferentes consejos (pieces-of- advice). De esto se sigue que la lista de consejos consiste de una lista ordenada de consejos .

El primer consejo que es satisfecho será ejecutado. Finalmente llegaremos a la etapa donde la definición de consejo , la idea central de AL1, será dada.

Un consejo (piece-of-advice) consiste de 5 elementos:

Descripción del elemento	Nombre del elemento
- Hacia quién va dirigido el consejo	X
- Una meta a realizar	better goal
- Ciertas condiciones a ser preservadas	holding goal
- Un subconjunto ordenado de movimientos legales del bando X	move constraints for X
- Un subconjunto ordenado de movimientos legales del bando Y	move constraints for Y

Nuevamente con el ejemplo de ajedrez uno podría sugerir un consejo para intentar dar jaque mate, suponiendo que la segunda precondición ha sido satisfecha. El consejo para las blancas es:

better goal	dar jaque mate
holding goal	no perder la torre y evitar tablas
move constraints para bando X	intentar solo movimientos de rey o jaque
move constraints para bando Y	evitar estar en línea con la oposición del rey

Recordemos, sin embargo, que dependemos de una lista de consejos y no sólo de un consejo, por lo tanto no hay necesidad de garantizar la obtención del better goal con este consejo, pero en caso de conseguirlo, decimos que el consejo se satisface.

2.2 Definición de los elementos centrales de AL1.

A continuación se describen las definiciones básicas que utiliza AL1 como se presentan en [Bratko 81].

Consejo (piece-of_advice).

Un consejo es un quíntuple (X,BGx,HGx,MCx,MCy) donde

X es el bando al cual está dirigido el consejo

BGx y HGx son predicados acerca de posiciones y

MCx y MCy son predicados acerca de movimientos.

BGx le llamamos better goal.

HGx " " holding goal.

MCx " " move constraints para "x".

MCy " " move constraints para "y".

Arbol forzado (Forcing tree).

Dada una posición Pos y un consejo A=(BGx,HGx,MCx,MCy) un árbol forzado para A en Pos es un subárbol T del árbol de la partida con raíz en Pos tal que:

1 Para todo nodo q en T: HGx(q);

2 Para todo nodo noterminal q en T: not BGx(q);

3 Para todo nodo terminal q en T:BG q) o q es un movimiento para "y: donde no hay movimiento legal que satisfaga MCy;

4 Hay exactamente un movimiento en T para cada movimiento de x en un nodo no terminal en T que debe satisfacer MCx;

5 Todos los movimientos legales de cualquier posición donde mueva "y" en T que satisfagan MCy están en T.

Satisfiable (satisfactibilidad)

Un consejo A es satisfactible en una posición Pos si existe un árbol forzado para A en pos. Escribimos sat(A,Pos) .

Plan.

Un plan P, es un cuádruple (X,BGX,HGX,MCX) donde X,BGX,HGX y MCX se interpretan como en el consejo. La única diferencia formal entre los conceptos de plan y consejo es que en los planes el oponente de X, o sea Y, no tiene restricción de movimiento

Exito de un Plan.

Un plan P=(X,BGX,HGX,MCX) tiene éxito en una posicion Pos si y solo si sat(A,Pos), donde A=(X,BGX,HGX,MCX, cualquier movimiento).

En este caso escribimos Suc(P,Pos) .

El 'move constraint' Y referido como cualquier movimiento se refiere precisamente a permitir cualquier movimiento legal para el bando Y.

2.3 Ejemplos

Ejemplo 1:

Tomando el ejemplo precedente de ajedrez dar mate con rey y torre contra rey, daremos un lenguaje de consejos en una forma más detallada.

Tabla de consejos.

En este caso los consejos son cuádruples (N,B H M) donde

- N — Número de movimientos hacia adelante
- B — Better goal
- H — Holding Goal
- M — Move constraint

En este caso se asume para quién es el consejo (quien tenga la torre) y se agregó N por motivos prácticos. Permitiremos una forma normal disyuntiva para definir B H y M en términos de predicados básicos.

Recuérdese que una tabla de consejos es una lista de reglas.

```
REYESOP ->
                    1                                   (N)
                    REYESOP                             (B)
                    PROTTORRE                           (H)
                    JAQUE                               (M)

TRDIVIDE ->
                    1
                    RESTRINGE
                    PROTTORRE TDIVIDE NOAHOGADO
                    MTORRE

TDIVIDE REYESLJ ->
                    1
                    REYESPC
                    RESTRIGUAL PROTTORRE NOAHOGADO
                    MREYB

TDIVIDE REYESLJ ->
                    1
                    MENORDIS
                    RESTRIGUAL PROTTORRE NOAHOGADO
                    MREYB

TDIVIDE  ->
                    1
                    LEJOSTORRE
                    RESTRIGUAL
                    MTORRE
```

CIERTO ->

2

TDIVIDE

PROTTORRE NOAHOGADO

MTORRE

De acuerdo a las definiciones dadas esta tabla utiliza planes en vez de consejos. Esta tabla no origina movimientos óptimos pero sí logra el fin de dar jaque mate.

Pueden ser encontradas tablas simples de consejos en la literatura [Bratko 78], pero es útil que cada quien intente definir su propia tabla de consejos.

A continuación daremos una explicación de los predicados usados en la tabla.

Nombre del predicado.	Definición.
TDIVIDE	Verifica que la torre divida al tablero en dos regiones hallándose los reyes en regiones distintas (Figura 1).
TRDIVIDE	Es un caso particular de TDIVIDE (figura 2).
REYESOP	Verifica que los reyes estén opuestos. Es decir que los reyes estén bien en un mismo renglón o una misma columna separados por una casilla libre.
REYESPC	Verifica que los reyes estén separados por un movimiento de caballo.
REYESLJ	NOT(REYESOP OR REYESPC)
LEJOSTORRE	Si la torre está lejos del rey negro. A más de 4 casillas de distancia.

NOCERCATORRE Si la torre y rey negro no se encuentran a una casilla de distancia.

JAQUE Si el rey negro está en jaque.

NOAHOGADO Si el rey negro no está ahogado.

PROTTORRE Si la torre está a salvo.

Los siguientes predicados establecen alguna relación entre 2 posiciones, Pi y Pj. TIpicamente Pj es una una posición posterior a Pi en el árbol de variantes.

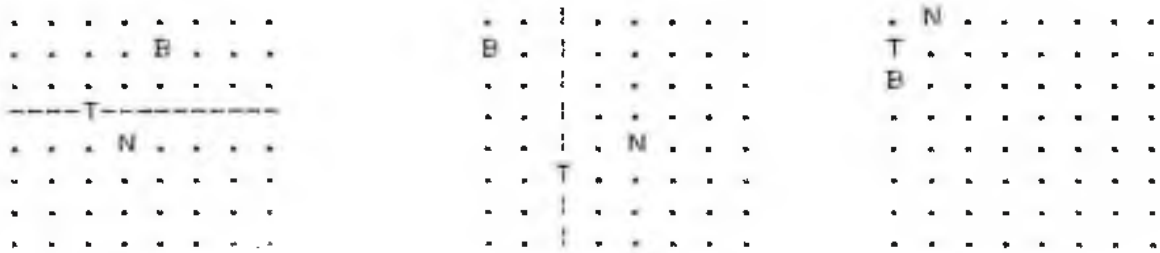
RESTRINGE LIBERTAD(Pj) < LIBERTAD(Pi) (figura 3 vemos l perta

RESTRIGUAL LIBERTAD(Pj) = LIBERTAD(Pi)

Predicados sobre movimientos

NTORRE Movimiento de torre

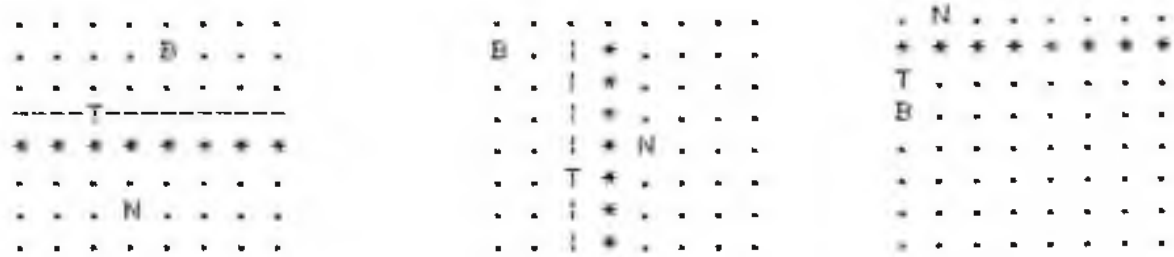
MREYB Movimiento de rey blanco



(a) (b) (c)

Fig. 1 (Ejemplo de TDIVIDE)

B es rey blanco , N es rey negro y T es torre blanca



(a) (b) (c)

Fig.2 (Ejemplos de TDIVIDE)

Libertad(Fig. 1a) = 5

Libertad(Fig. 1b) = 6

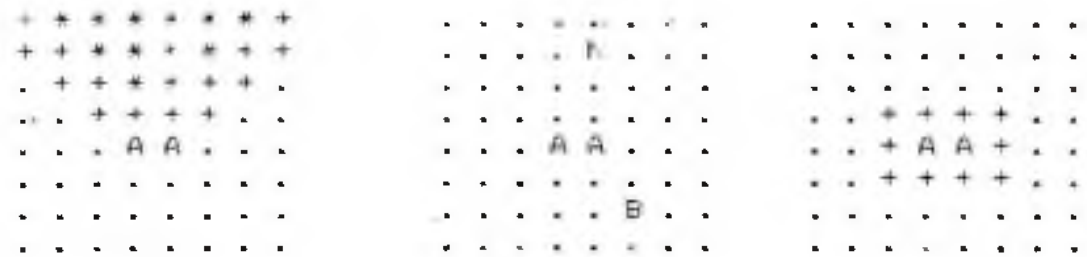
Libertad(Fig. 1c) = 2

Libertad(Fig. 2c) = 3

Fig. 3 (Valores de la función de libertad para los ejemplos de arriba)

Ejemplo 2:

El siguiente ejemplo tiene el objetivo de dar jaque mate con 2 alfiles. Supondremos que el blanco es quien posee a los alfiles. Este problema aunque no puede considerarse difícil en ajedrez es más interesante que el mate con torre visto previamente. Inicialmente daremos una explicación de las bases del procedimiento que se utilizará.



(a) (b) (c)

Figura 4

En la figura 4a llamemos a la región enmarcada con asteriscos "campo de control de los alfiles". Esto es porque si el rey negro se halla situado en esta región no tiene forma de salir de ella. Los alfiles atacan directamente a los escaques señalados con el signo de +. Llamemos a esta región, frontera. Finalmente a la región de puntos la llamaremos región externa . Estas definiciones son sobre uno de los campos de control de los alfiles, pues siempre hay 2 de éstos. En esta figura corresponden al que se halla en la parte superior de los alfiles y el que se halla en la parte inferior de los alfiles. Pero en una posición concreta solo nos interesará uno. Precisamente donde se encuentre el rey negro. Asi pues definimos el patrón de posiciones lámpara_alumbra como : Cualquier posición donde los alfiles estén juntos generando un campo de control en donde se halle el rey negro y que el rey blanco se halle en la región externa a dicho campo. La figura 4b es un caso particular de dicho patrón.En la figura 4c definimos la región señalada con el signo + como región cerca de la lámpara. Definimos el patrón blanco_cerca_ _lámpara si en una posición el rey blanco se halla en la región cerca de la lámpara.

Los nombres utilizados para los patrones intentan hacer un símil con el campo de luz generado con una lámpara de mano o linterna. usando más de este símil podemos llamar al campo de control de los alfiles el campo de luz de la lámpara. Ahora bien, si lanzamos una lámpara de éstas a una pared podemos notar que disminuye este campo de luz reflejado sobre la pared. Por tanto para disminuir el area de control de los alfiles donde se

encuentra el rey negro, es necesario acercar la lámpara hacia el rey negro. Esto se hace con ayuda del rey blanco. Dicho de otro forma, el rey blanco es quien se encarga de mover la lámpara.En la figura 5 se muestra un ejemplo de una secuencia de acciones mediante las cuales el rey blanco mueve la lámpara hacia el rey negro, restringiendolo a un campo de luz menor.

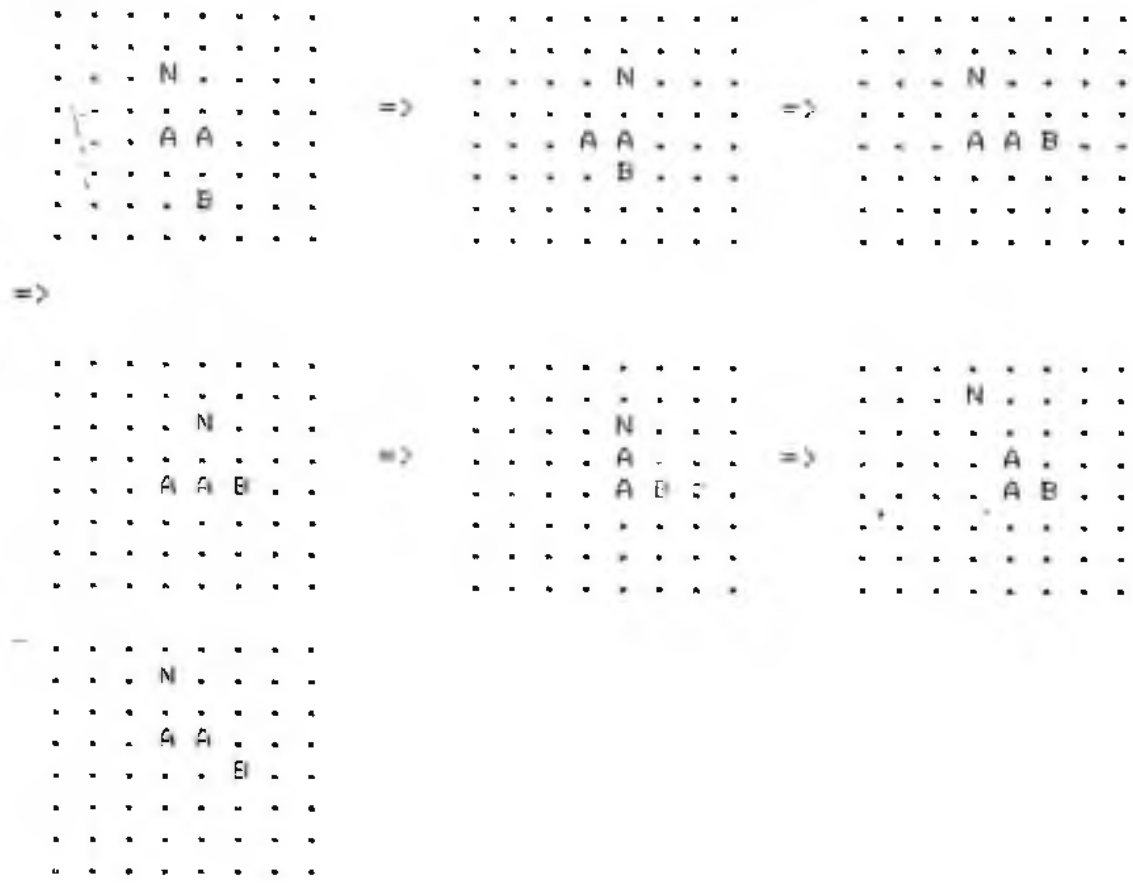


Figura 5

Así, es posible orillar al rey negro. Es decir llegar a una posición como la mostrada en la figura 6a

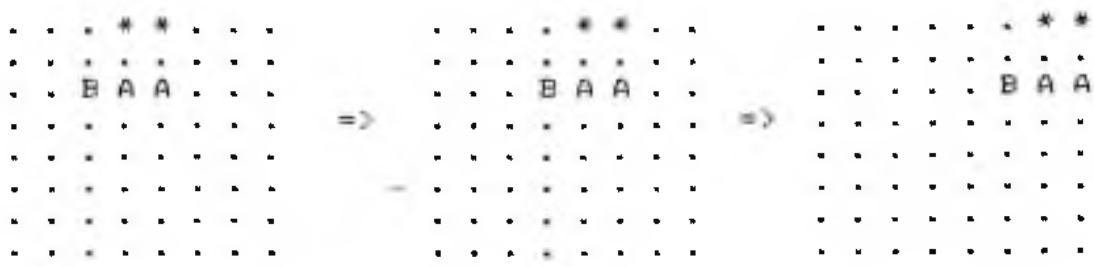


Figura 6

donde el rey negro se halla en la región de control señalada.

De la figura 6a se moverá la lámpara a la figura 6b hasta que finalmente se llegue a la figura 6c. Nuevamente podemos decir que el rey blanco es quien hace el movimiento de la lámpara.En la figura 7 se muestra un ejemplo de una secuencia de acciones que permite hacer el movimiento de lámpara de 6a en 6b. En esta figura se muestran únicamente los movimientos del blanco.

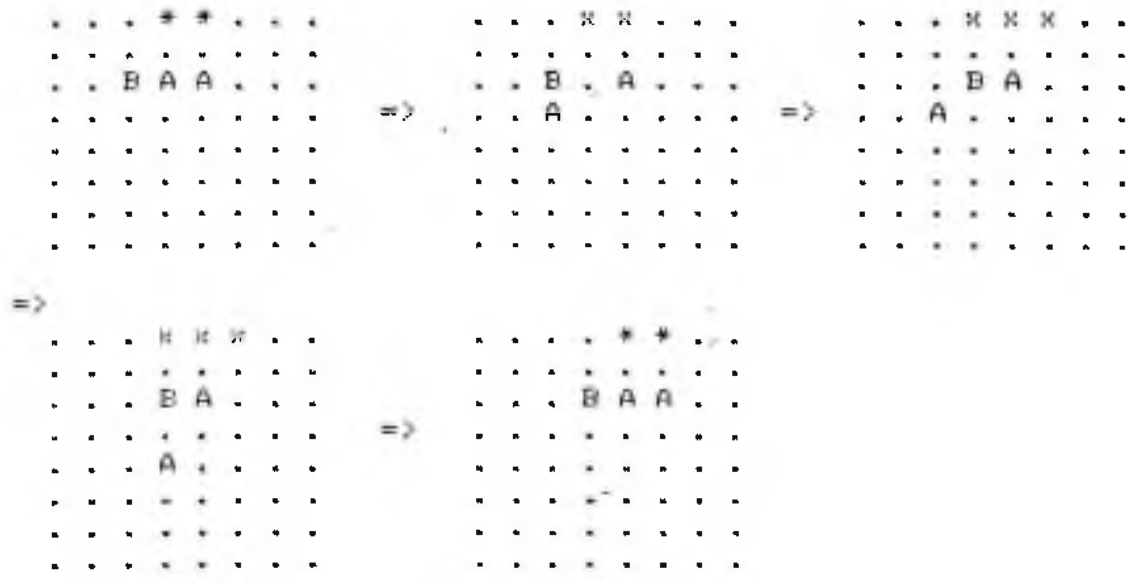


Figura 7

En la figura 7 que se acaba de presentar se usan asteríscos para señalar el área de control inicial y final y se usan equis para indicar las posibles casillas del rey negro.

Esta idea de orillar y después esquinar al rey negro por medio de la lámpara para finalmente matarlo se expresa mediante los siguientes consejos :

```
lámpara_alumbra_esquina ->
                          4
                          jaque_mate
                          noahogado alfiles_a_salvo
                          malfil1 o mreyb

lámpara_alumbra_orilla  ->
                          4
                          lámpara_alumbra_orilla
                          desplaza_campo
                          noahogado alfiles_a_salvo
                          malfil1 o malfil2 o mreyb


lámpara_alumbra  blanco_no_cerca_de_lámpara  ->
                          1
                          aproxima_blanco_a_lámpara


                          mreyb
```

```
lámpara_alumbra blanco_cerca_de_lámpara      ->
                                4
                                reduce_campo   o
                                lámpara_alumbra_orilla    noahogado
                                alfiles_a_salvo
                                malfil1 o mreyb

no_lámpara_alumbra           ->
                                4
                                lámpara_alumbra
                                alfiles_a_salvo noahogado
                                -
```

A continuación se dará una breve explicación de los predicados usados en la tabla de consejos.

Nombre del predicado	Definición
lámpara_alumbra	como se explicó anteriormente
no lámpara_alumbra	patrón complemento a lámpara_alumbra
lá ara_alumbra_orilla	lampara_alumbra como en fig 6a y ey blanco cerca de la lampara
lá para_alumbra_esquina	lám a_alumbra com en fig 6c y ey blanco cerca de la lámpara
alfiles_a_salvo	que ningún alfil pueda ser capturado
malfil1	alfil se mueve un solo cuad o
malfil2	alfil se mueve dos cuadros
mreyb	movimiento de rey blanco
desplaza_campo	el campo se desplaza como cambio de figura 6a a 6b
reduce_campo	cuando la lámpara se aproxima al negro disminuyendo el campo de luz
blanco_cerca_de_lámpara	ya se explicó

Generalización de ambos ejemplos.

Ambos problemas (mate con torre y mate con alfiles) siguen un mismo patrón de solución que es el siguiente:

1.- Búsquese un cierto patrón ideal de posisiones tal que el rey negro este confinado a una región del tablero de la cual no pueda salir.

2.- Búsquese una secuencia de movimientos que puedan restringir esta región del rey negro. Se sugiere utilizar al rey blanco para este próposito y por tanto parece ser útil tenerlo cerca de esta región.

3.- Cuando el rey negro se halle totalmente confinado (en una orilla o quizá incluso esquina del tablero), búsquese una secuencia de operaciones que puedan dar mate al rey.

Si pudieramos formalizar más estas ideas (quizá en términos de consejos) podríamos tener un mecanismo que 'halle' precisamente la tabla correspondiente de consejos necesaria para llegar a la meta deseada (mate en este caso). En términos de consejos podríamos intentar definir una tabla de consejos general que halle una tabla de consejos para un problema propuesto que pertenezca a esta familia de problemas. Esta podría ser una linea abierta de investigación.

2.4 El Sistema AL1.

AL1 consiste de cuatro módulos independientes:

1) Resolvedor del problema. Ejecuta la búsqueda en todo el espacio del problema por medio de una función de movimientos legales, usando el conocimiento del problema específico dado en la tabla de consejos.

2) Editor interactivo de la tabla.

   Es usado para crear la tabla de consejos.

3) Módulo del juego. Ejecuta la estrategia generada por el resolvedor del problema en la forma del árbol forzado (Huberman type).

   El forcing tree nos dice cuando un consejo está satisfecho y cómo ejecutarlo.

4) Base de predicados. Está formada por un conjunto de predicados que caracterizan a un problema particular. Los predicados constituyen a los consejos así como las precondiciones d éstos.

2.5 Conclusiones generales sobre AL1.

Ventajas de AL1

Claridad.

Observe que el código de una tabla de consejos es fácilmente entendido y cualquiera puede usarlo para aprender directamente de éste en una manera sistemática, explícita y precisa.

Simplicidad.

Es fácil crear una tabla de consejos en tanto que tengamos nuestras ideas ordenadas de cómo resolver el problema. No hay necesidad de programar en el sentido tradicional: uno sólo necesita especificar las ideas.

La capacidad para dividir el problema en subproblemas.

AL1 hace posible esto al descomponer la ejecución del problema en una última meta que consiste en una secuencia de submetas, cada una de ellas correspondiendo a un better goal de algún consejo [Bratko 81].

Susceptibilidad de pruebas formales de la correctitud de estrategias.

Parece ser que los puntos de arriba justifican de una manera natural esta proposición. -Sin embargo, el lector interesado es referido a [Bratko 1978] .

Niveles de Abstracción en AL1

Uno puede encontrar los siguiente dos niveles de abstracción:

- Selección del predicado base que conforma la tabla de consejos.
- La estructura de la tabla de consejo.

Esta separación parece ser adecuada en ajedrez, donde uno puede definir ciertos parámetros que caracterizarán las posiciones (base de predicados) y entonces definir ciertas relaciones entre éstos para poder expresar propiedades de la posición. Para ser más explícito ,suponemos un final de peones. En este caso no es difícil suponer que los siguientes parámetros determinan la posición:

Ventaja material, promoción de peones ,peones doblados, oposición

de los reyes, etcétera.

Y en otro nivel de abstracción debemos conocer cómo se relacionan estos parámetros para ganar el juego. Este segundo nivel de abstracción, más difícil de definir, determina la estructura de la tabla de consejos.

Prolog y AL1.

AL1 puede ser entendido como una metodología de programación en Prolog. Por ejemplo podemos tener un predicado de Prolog con 3 argumentos:

Nombre de la tabla de consejos (t).

-Situación del problema, es decir, una posición de ajedrez (p).

-Siguiente situación, es decir, movimiento sugerido(m).

Este predicado será satisfecho cuando la tabla de consejos (t) en la posición del problema (p) sugiere el movimiento (m).

Si uno hace un intérprete en Prolog de AL1 consistirá principalmente de un verificador de los predicados mencionados anteriormente. En este sentido, ambos conceptos, metodología e intérprete tienen el mismo significado.

Sin embargo, por razones de ejecución,podemos desarrollar una implementación particular de AL1 (en un lenguaje apropiado) y usarlo vía Prolog. Alguien podría tener en mente un lenguaje diferente de Prolog y observar que todo lo mencionado anteriormente puede aplicarse igualmente. Sin embargo las cualidades mencionadas de AL1 son también compartidas por Prolog y no por cualquier lenguaje . Desde un punto de vista conceptual tienen una estrecha relación.

Es interesante notar que AL1 tuvo suficientes cualidades que le permitieron evolucionar a AL3, donde nuevos conceptos son encontrados tales como planes, métodos y lemas. Todos esos conceptos pueden ser manejados en Prolog sin muchas dificultades, pero la situación podría ser un tanto forzada, mientras que en AL3 se expresan con plena naturalidad .

-Posible reforma al concepto de árbol forzado:

En la definición de árbol forzado se pide que el holding-goal se satisfaga en todo nodo del árbol. Esto no permite transportar naturalmente el concepto de sacrificio en ajedrez. Parece interesante pedir que el holding-goal se satisfaga tan solo en los nodos no-terminales y así permitir el sacrificio de un holding-goal para alcanzar un better-goal. Lo que me parece especialmente interesante de esta redefinición de árbol forzado es que puede absorver fácilmente a la definición original mientras que el recíproco no se da. Expliquemos esto con más detalle:

Sea A= (Bgx,Hgx,Mcx,Mcy) un consejo original. Sea sat(A,Pos) la definición de satisfactibilidad original. Y sea sat_n(A,Pos) la definición nueva de satisfactibilidad (será la misma pero con la nueva versión de árbol forzado). Evidentemente sat(A,Pos) => sat_n(A,Pos) pero el recíproco no necesariamente es válido. Pero si queremos lograr el mismo efecto original podemos transformar autómaticamente el consejo A al consejo

A1=(Bgx & Hgx,Hgx,Mcx,Mcy) y entonces sat_n(A1,Pos) cumple precisamente con la definición de sat(A,Pos) y esta tranformación fue simple.

En cambio no parece fácil hallar automáticamente A a partir de : A1=(Bgx,Hgx,Mcx,Mcy) tal que sat(A,Pos) <-> sat_n(A1,Pos)

Implementaciones:

A continuación presento una breve explicación de cada implementación hecha (en Micro-Prolog y Turbo-Pascal) :

- Implementación en Micro-Prolog.

Mostraremos tan solo la clausula que verifica si un consejo se satisface:

Argumentos:

X7	Posición inicial
X1	Posición corriente
X2	Posición después de X1 según movimiento legal
x	Cota de movimientos
X	Better-Goal
Y	Holding-Goal
Z	Move Constraint para quien va dirigido el consejo
Z1	Move Constraint del adversario

```
((GOR X7 X1 (x X Y Z Z1) X2)
     (MOVLEGAL X1 X2)
     (SATISF_FUN_LOG Z (X7 X1 X2))
     (SATISF_FUN_LOG Y (X7 X1 X2))
     (OR ((SATISF_FUN_LOG X (X7 X1 X2) ))
         ((LESS 1 x)(GAND X7 X2 (x X Y Z Z1) X3) )
     )
))

((GAND X7 X1 (x X Y Z Z1) X2)
     (SUM x1 1 x)
     (FORALL ((MOVLEGAL X1 X2)
          (SATISF_FUN_LOG Z1 (X7 X1 X2)))
          ((GOR X7 X2 (x1 X Y Z Z1) X3))
))
```

Una discrepancia de esta definición en Prolog con la definición original es que no verificamos Better-Goal ni Holding-Goal en el turno del adversario. Esta particularidad es intencional considerando que es poco probable llegar al better-goal en el turno del adversario y en cambio es costoso el verificarlo. Analogos motivos son para el holding-goal.
La clausula SATISF_FUN_LOG verifica que se cumpla una condición lógica.

Implementación en Pascal:

La versión de AL1 en Prolog resulta especialmente útil para investigar las posibilidades de expansión del lenguaje, pero puede ser desesperante ver funcionar al sistema en un problema concreto. El problema de mate de los alfiles expuesto anteriormente funcionó tan solo parcialmente en Micro-Prolog por motivos de falta de espacio. El consejo encargado de dar jaque mate que se presenta cuando se satisface la precondición lámpara_alumbra_esquina requería varios minutos, por tanto se decidió hacer una implementación de Al1 en Pascal para poder ofrecer al usuario un producto atractivo. Se trabajo sobre el ejemplo de mate de alfiles y se incrementó el factor de velocidad de minutos a segundos. La implementación hecha puede optimizarse aún más,pues con la intención de mostrar algunos algoritmos claros, se perdió eficiencia en la implementación. A su vez se aprovechó un procedimiento en Pascal que permite desplegar un tablero de Ajedrez elegante y hacer en general la entrada y salida del programa al realizar movimientos es amigable.

Algunos aspectos técnicos de la implementación:

Se utilizó Turbo-Pascal versión 3.0 y se ejecutó en una microcomputadora BPM/XT PC compatible bajo el sistema operativo MSDOS versión 3.20. Se implementó un traductor de AL1 a Pascal.

En vez de utilizar consejos, se utilizan planes. El motivo es que me resultó siempre más sencillo definir estrategias a partir de planes en vez de a partir de consejos. El hecho de que muchos ajedrecistas compartieran este punto de vista conmigo reforzó mi

decisión.

La parte fundamental del programa es la de verificar si un plan se satisface, por tanto presentó el codigó en Pascal para este efecto y lo discuto a continuación.

```
function  advice_y1(cota:integer;pos:posicion;
                    var pos1:posicion):boolean;forward;

function advice_1(cota:integer; pos:posicion;var p s1:posicion):bool a
var
  numero_mov : integer;
  pos2       :posicion;


function better_goal(pos:posicion):boolean;

begin
  better_goal:=jaque_mate(pos);
end;

function holding_goal(pos:posicion):boolean;

begin
  holding_goal:=(not ahogado(pos)) and (alfiles_a_salvo(pos));
end;


function mcx(pos,pos1:posicion):boolean;

begin
  mcx:=(mreyb(pos,pos1) or malfil1(pos,pos1));
end;
```

```
begin
     numero_mov:=0;
     while legal_move_x(numero_mov,pos,pos1) do
       begin
          if mcx(pos,pos1) and holding_goal(pos1) then
          begin
             if better_goal(pos1) then
               begin
                advice_1:=true;
                exit;
               end

              else
              if advice_y1(cota-1,pos1,pos2) then
               begin advice_1:=true;exit;end
              else advice_1:=false;
           end;
      end;
end;

function advice_y1;
var
  numero_mov :integer;
  pos2       :posicion;

begin
     numero_mov:=0;
     if cota>0 then
        begin
        while legal_move_y(numero_mov,pos,pos1) do
          begin
             if not advice_1(cota,pos1,pos2) then
                begin
                  advice_y1:=false; exit;
                end;
          end;
          advice_y1:=true;
          end
     else advice_y1:=false;
end;
```

Esta versión es análoga a la presentada en Micro-Prolog, en el sentido de que es recursiva y por tanto la búsqueda es a lo profundo y existe una cota en la altura del árbol a buscar. El better-goal y holding-goal solo se verifican después del movimiento de x (x es a quien esta dirigo el plan). Al no contar con la posibilidad del backtraking de Prolog debemos realizar un ciclo hasta hallar algún movimiennto de x que satisfaga el plan (si lo hay ?) y utilizamos una función que trae movimientos legales, la cual utiliza un argumento extra (numero_mov) para recorrer todo los posibles movimientos legales.

Finalmente, las funciones para better_goal, holding_goal y mcx se tienen para un plan concreto, pero pueden definirse adecuadamente para cada plan. Utilizando el lenguaje C u otra versión de Pascal tales funciones podrían ser argumentos de la funcion advice_1.

3. AL3

3.1 Definición de AL3

Vista General de AL3.

Desde un punto de vista lógico, el sistema es un demostrador automático de teoremas del ajedrez. Si se le pide probar P el sistema intenta:

Demostrar P o

Demostrar no P o

Hallar que el valor de P es indeterminado

Por otro lado, de acuerdo a su presentación es un sistema experto, pues esta formado por :

Una base de conocimientos (métodos, planes, lemas, hechos, hipótesis)

Un esqueleto o módulo de control muy general que soluciona problemas según su base de conocimientos.

3.1.1 Antecedentes.

AL1.5

AL1 evolucionó hacia nuevas versiones. El primer paso fue AL1.5 en donde se agregan las siguientes posibilidades sobre AL1 [Bratko y Niblett 79]:

a) Llamadas a consejos dentro de los mismos consejos. Esto permite recursividad en los consejos.

b) El usuario puede definir la salida de una búsqueda cuando no hay movimiento legal que satisfaga tanto al 'move constraiht' como al 'holding goal' .

c) Se permiten búsquedas semiestáticas cuando se consideran movimientos sólo de algún bando.

d) Los 'move constraints' son aplicables a cualquier nivel de profundidad en el árbol de búsqueda (en AL1 se permitían sólo en el primer nivel).

AL2

Posteriormente surgió AL2 con ligeras mejoras, pero esto sistemas mantienen en esencia la misma estructura de control de AL1.

Restriciones compartidas por AL1, AL1.5, AL2

En el ajedrez se requiere todavía manejar un mayor número de conceptos de alto nivel . Un ejemplo es el concepto de 'refutación'. Si el diseño de AL1 es apropiado para las pruebas formales de correctitud de estrategias no lo es en cambio para probar la incorrectitud de las estrategias.

Por medio del uso de 'refutaciones' se logra este cometido. Por otro lado, en AL1 el uso de consejos es estático. Es decir, el usuario alimenta al sistema de un conjunto de consejos y la capacidad del sistema para solucionar un problema está sujeta a este conjunto predefinido. Sería conveniente que el sistema pudiera 'crear' consejos de alguna forma. Como se observa esta evolución fue en un sentido técnico mas no conceptual.

3.1.2 Elementos de AL3.

Hipótesis :

Son conjeturas sobre las propiedades del problema a resolver. Estas conjeturas son las ideas que se tienen del problema y que AL3 determinará su valor de verdad.

Denotaremos con Hn a las hipótesis, donde n es un numero

natural. Un ejemplo simple de hipótesis es :

HO : Puede ganar el bando blanco ?

Hechos :

Conforme las hipótesis se investigan, se obtienen sus valores de verdad. Así las hipótesis se convierten en hechos (falsos o verdaderos). Por ejemplo, una vez que el sistema examine la hipótesis HO definida antèriormente, puede concluir que es cierta. En ese momento, la hipótesis HO se convierte en un hecho verdadero.

Los axiomas del problema, también son hechos. Por ejemplo en ajedrez un hecho o axioma es :

El blanco gana si y solo si el negro pierde.

Métodos:

Para investigar los valores de verdad de las hipótesis se usan los métodos. Mas formalmente, dada una hipótesis, un método puede considerarse como una caja negra que obtiene alguna de las siguientes respuestas:

a) Establece que la hipótesis es cierta.

b) Establece que la hipótesis es falsa.

c) Establece que la hipótesis es falsa o cierta según el valor de verdad de otras hipótesis.

d) Establece que no puede obtener una conclusión sobre el valor de verdad de la hipótesis.

Un ejemplo de método en un final de peón y rey contra rey en ajedrez es : Encuentra que el partido se gana verificando que es

posible coronar.

Considerando al método como un procedimiento según se definió previamente, al sistema se le presentará la tarea de averiguar si en una posición dada es posible ganar. El sistema entonces encuentra en su base de conocimientos el hecho (o axioma) :

Ganar es equivalente a Coronar

y por tanto responde (de acuerdo al inciso c) que el valor de verdad de la hipótesis Ganar es equivalente al valor de verdad de la hipótesis Coronar.

Asociado a cada método existen una serie de precondiciones que deben cumplirse para que éste pueda activarse.

Lemas:

Son un caso especial de métodos . Estos carecen de los incisos c) y d) definidos para los métodos. Corresponden al concepto de teoremas sobre el problema dado. Un ejemplo de lema es la regla del cuadrado que ya fue mencionada en 1.4.2. (ver apéndice).

Consejo (Piece-of-advice).

Un consejo es un quintuple (X,BGx,HGx,MCx,MCy) donde

X es el bando al cual está dirigido el consejo

BGx y HGx son predicados acerca de posiciones y

MCx y MCy son predicados acerca de movimientos.

BGx le llamamos better goal.

HGx " " holding goal.

MCx " " move constraints para "x".

MCy " " move constraints para "y".

Árbol forzado (Forcing tree).

Dada una posición pos y un consejo A=(BGx,HGx,MCx,MCy) un árbol forzado para A en pos es un subárbol T del árbol de la partida con raíz en pos tal que:

1 Para todo nodo q ∈ T: HGx(q); (se mantienen las condiciones en todo el arbol)

2 Para todo nodo noterminal q en T: not BGx(q); (se realiza el objetivo sólo en nodos terminales)

3 Para todo nodo terminal q en T:BGx(q) o q es un movimiento para "y: donde no hay movimiento legal que satisfaga MCy; (realización del objetivo)

4 Hay exactamente un movimiento en T para cada movimiento de x en un nodo no terminal en T que debe satisfacer MCx; (sólo se considera un movimiento por nodo)

5 Todos los movimientos legales de cualquier posición donde mueva "y" en T que satisfagan MCy estan en T (se consideran todas los movimientos de y).

Satisfactibilidad (Satisfiable)

Un piece-of-advice A es satisfactible en una posición Pos si existe un árbol forzado para A en Pos. Lo denotamos con sat(A,Pos). Una propiedad interesante que se desprende de las definiciones anteriores es :

sat((X,BGX,HGX,MCX,MCY),Pos) si y solo si

not sat((Y,not(HGX),not(BGX and HGX),MCY,MCX),Pos)

A esta relación se le conoce como 'la relación inversa del consejo. Aqui el 'better goal' de Y consistirá en que no se realice el HGx en algún momento. Aplicando el operador negación en la relación del 'holding goal' de Y se observa que con esto pretende evitar el BGx o evitar el HGx.

Plan.

Un plan P, es un cuádruple (X,BGX,HGX,MCX) donde X,BGX,HGX y MCX se interpretan como en el consejo. La única diferencia formal entre los conceptos de plan y consejo es que en los planes el oponente de X, o sea Y no tiene restricción de movimiento.

Exito de un Plan.

Un plan P=(X,BGX,HGX,MCX) tiene éxito en una posición Pos si y solo si sat(A,Pos), donde A=(X,BGX,HGX,MCX,cualquier movimiento)

En este caso escribimos Suc(P,Pos)

El 'move constraint Y referido como cualquier movimiento se refiere precisamente a permitir cualquier movimiento legal para el bando Y.

Refutación.

Sean Px y Py los siguientes planes:

Px = (X,BGX,HGX,MCX)

Py = (Y,BGY,HGY,MCY)

entonces se dice que el plan Py refuta al Plan Px en una posición Pos y se denota como ref(Py,Px,Pos) si y solo si sat(A,Pos) donde:

A= (Y,not(HGX), not(HGX) or not(BGX) and HGY,MCY,MCX).

Es decir ref(Py,Px,Pos) <=> sat(A,P)

Utilizando la propiedad de la relación inversa del consejo se tiene que ref(Py,Px,Pos) <=> not sat(A1,Pos) donde

A1 = (X,BGX or not (HGY),HGX,MCX,MCY)

La propiedad esencial de una refutación es que :

ref(Py,Px,Pos) => not suc(Px,Pos)

Nótese que el concepto de refutación no exige que Py tenga éxito, es decir se alcance su 'better goal' sino que se preocupa por destruir las posibilidades de éxito de Px.

Obsérvese que puede ser muy costoso verificar el éxito o fracaso de un plan por medio de su traducción a consejo, ya que no hay restricción de movimientos para uno de los bandos . Sin embargo

por medio de refutaciones se obtiene la ventaja de que ambos bandos tienen restricción de movimientos y por tanto la búsqueda del árbol forzado será más económica. Esta forma es ventajosa para probar que un plan no puede tener éxito.

Utilidad de estos conceptos en Ajedrez.

La importancia de los conceptos 'consejo' y 'plan' es dada por los capítulos anteriores, tales conceptos tienen un uso algo restingido en AL1, ya que en Ajedrez es usual buscar 'contraplanes' es decir ideas o recursos que invaliden el uso de algún plan. El término más adecuado para denotar este concepto es precisamente el de 'refutación', término ampliamente conocido y utilizado en Ajedrez. La existencia de este término le da un mayor dinamismo al mismo concepto de plan.

En cuanto a los métodos, su importancia va más allá del ajedrez, pues toda disciplina científica y en general casi cualquier actividad humana requiere métodos. Su principal uso en este contexto es el de darles a las estrategias un orden lógico e indicar en qué situaciones específicas se aplica cada una de ellas.

3.1.3 AL3 como un sistema experto.

Puede considerarse a AL3 como un sistema experto, pues cuenta con esqueleto o módulo de control y una base de conocimientos para resolver los problemas propuestos, como previamente se hizo notar.

Funcionamiento del módulo de control.

El proceso de solución de problemas se hace mediante una secuencia de ciclos de ejecución, esto es supervisado por el módulo de control.

Un ciclo de ejecución realiza la siguiente labor:

Analiza si ya se tiene una respuesta a la pregunta inicial. En caso afirmativo se da la respuesta hallada. En caso negativo, se busca qué método o lema puede utilizarse que ayude a responder la pregunta, aunque esto conduzca a generar más hipótesis por investigar. Para verificar que método puede aplicarse, se checa que se satisfagan sus precondiciones. Como puede ocurrir que más de un método pueda aplicarse ,la decisión se toma en base a un criterio de costo. Se espera seguir el método que utilice el camino más económico.

Una vez escogido el método, éste origina tener que actualizar la base de conocimientos con nuevos hechos y quizá nuevas hipótesis.La secuencia de ciclos de ejecución conduce eventualmente hacia alguna respuesta según el problema dado y la base de conocimientos.

Base de conocimientos.

Conviene separarla en:

Base de conocimientos (fija): Contiene los recursos (métodos, lemas, planes) con los que se espera poder solucionar los problemas propuestos. Corresponde a la base de conocimientos de entrada inicial proporcionada al sistema.

Se le llamará simplemente base de conocimientos.

Base corriente de conocimientos : Cambia según el transcurso de los ciclos de ejecución y contiene :

- Pregunta P del usuario (query) llamada **target** (objetivo). Como ya se dijo previamente, el sistema intentará probar P, o probar no P .
- Hipótesis del problema, las cuales se van agregando por medio de los metodos que se utilicen.
- Hechos que se van concluyendo de las hipótesis y que participan en la solución final del problema.
- Planes, consejos y otros objetos que son relevantes de alguna manera al procedimiento de resolución y que han sido generados durante la ejecución.

Ejemplo de una base de conocimientos

El siguiente ejemplo corresponde a una base conocimientos muy simple para un final de juego de ajedrez del tipo:

BLANCAS : Rey y Peón NEGRAS : Rey

Este ejemplo se relaciona con el punto 3.2.2, donde se explican los enunciados CORRE_PEON, CASILLAS_CRITICAS, y PATRON_BP6. Se pretende dar una idea de las sintaxis de la base de conocimientos La base se encuentra después de la siguiente explicación.

Comentario sobre la base ejemplo

La base inicia con OBSERVACIONES que es un elemento no presente en AL3, pero que por mi propia experiencia decidí introducir con la idea de que AL3 realice ciertas observaciones previas al proceso de demostración. En este caso concreto, AL3 determinará al inicio si el peón es de torre.

Hay dos tipos de métodos, fáciles y difíciles, lo cual da un criterio de costo. Escogiendo un método aleatoriamente se observan sus precondiciones, hipótesis H1, y la relación lógica con las acciones ,Hm m mayor que 1. Como se observa, H1 se producirá si sucede la relación lógica de Hm, para m mayor que 1. Para verificar lo anterior ahora Hm será la precondición de un método. Seguiremos este proceso hasta llegar al nivel más bajo de la base de conocimientos.

Los better goals y holding goals tienen el argumento P2 que se refiere a la posición descrita arriba. Los move constraints tienen dos argumentos P1 y P2 para denotar movimientos.

Los conectivos lógicos utilizados son :

+	disyunción
&	conjunción
~	negación
=>	implicación
<->	si y solo si

OBSERVACIONES (PEON_T)

LEMAS (PIERDE_PEON CORRE_PEON CASILLAS_CRITICAS CASILLAS_CRITICAS_T PEON_BLOQUEADO)

METODO MO FACIL

Precondiciones:

H1 : GANA

PEON_T = FALSO

Acción:

H2 : GANA_PEON_NT

Relación lógica entre las hipótesis

(H2 => H1)

METODO M1 FACIL

Precondiciones:

H1 : GANA

PEON_T = CIERTO

AcciÓN :

H2 : GANA_PEON_T

Relación lógica entre las hipótesis

(H2 => H1)

METODO M2 FACIL

Precondiciones:

H1 : GANA_PEON_NT

PEON_T = FALSO

Acción:

H2 : CORRE_PEON

H3 : CASILLAS_CRITICAS

H4 : PATRON_BP6

Relación lógica entre las hipótesis

((H2 o H3 o H4) => H1)

METODO M3 FACIL

Precondiciones:

H1 : GANA

AcciÓN:

H2 : TABLAS

Relación lógica entre las hipótesis

(H1 <-> ~ H2)

METODO M4 FACIL

Precondiciones:

H1 : TABLAS

Acción:

H2 : PIERDE_PEON

H3 : PEON_BLOQUEADO

Relación lógica entre las hipótesis

((H2 o H3) => H1)

METODO M5 DIFICIL

Precondiciones:

H1 : GANA_PEON_NT

Acción

H2 : SUC(P_BLANCO_NT)

Relación lógica entre las hipótesis

(H2 => H1)

METODO M6 DIFICIL

Precondiciones:

H1 : GANA_PEON_T

Acción

H2 : SUC(P_BLANCO_T)

Relación lógica entre las hipótesis

(H2 => H1)

```
METODO M7 DIFICIL

     Precondiciones:

        H1 :  SUC(P_BLANCO_NT)

     Acción

        H2 :  REF(P_NEGRO P_BLANCO_NT)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

METODO M8 DIFICIL

     Precondiciones:

        H1 :  SUC(P_BLANCO_T)

     Acción

        H2 :  REF(P_NEGRO P_BLANCO_T)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

METODO M9 DIFICIL

     Precondiciones:

        H1 :  TABLAS

     Acción

        H2 : SUC(P_NEGRO)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

PLAN P_BLANCO_T  (

                 BLANCAS

                 CASILLAS_CRITICAS_T (P2) + CORRE_PEON (P2)

                 PEON_A_SALVO (P2) & NOAHOGADO (P2)

                 MREYB (P1 P2) & RB_CERCA (P1 P2)

                )
```

```
METODO M7 DIFICIL

     Precondiciones:

        H1 :  SUC(P_BLANCO_NT)

     Acción

        H2 :  REF(P_NEGRO P_BLANCO_NT)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

METODO M8 DIFICIL

     Precondiciones:

        H1 :  SUC(P_BLANCO_T)

     Acción

        H2 :  REF(P_NEGRO P_BLANCO_T)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

METODO M9 DIFICIL

     P  condiciones:

        H1 :  TABLAS

     Acción

        H2 : SUC(P_ EGRO)

                Relación lógica entre las hipótesis

                 (H2 => ~ H1)

PLAN P_BLANCO_T  (

                 BLANCAS

                 CASILLAS_CRITICAS_T (P2) + CORRE_PEON (P2)

                 PEON_A_SALVO (P2) & NOAHOGADO (P2)

                 MREYB (P1 P2) & RB_CERCA (P1 P2)

                )
```

```
PLAN P_BLANCO_NT  (
                  BLANCAS
                  CASILLAS_CRITICAS (P2) + CORRE_PEON (P2) +
                           PATRON_BP6 (P2)
                  PEON_A_SALVO (P2) & NOAHOGADO (P2)
                  MREYB (P1 P2) & RB_CERCA (P1 P2)
                 )

PLAN P_NEGRO (
                  NEGRAS
                  PIERDE_PEON (P2) + PEON_BLOQUEADO (P2)
                  no hay
                  RN_CERCA (P1 P2)
```

3.1.4. Metaconocimiento sobre planes en AL3.

En AL3 es posible crear planes a partir de otros, según ciertos motivos observados de la posición. Cuando el conjunto de planes y consejos definidos inicialmente es incapaz de obtener alguna solución al problema se usan dos principios básicos para generar planes más elaborados a partir de otros [Bratko 84] :

a) Combina dos planes que se sabe que fallan por medio de una conexión 'or'.

b) Para un plan que falla, encuentra el motivo de la falla y modifica este plan de tal manera que se espere que se elimine el motivo de la falla.

Explicación más detallada de los metaplanes.

Metaplan OR

Supongamos que se tiene la siguiente situación:

La hipótesis H : Suc(P,Pos)

Para investigarla se intentó refutarla por medio de planes R1 y R2. Ambos por su cuenta fallaron, es decir se tienen los hechos:

H1 : Ref(R1,P,Pos) es falso y H2 : Ref(R2,P.Pos) es falso entonces porqué no juntar fuerzas entre R1 y R2 para refutar a P (juntos venceremos). Pues bien, precisamente esta es la acción tomada por el metaplan OR, es decir se genera la acción:

Hipótesis H3 : ref(R1 or R2 ,P , Pos) con la relación lógica

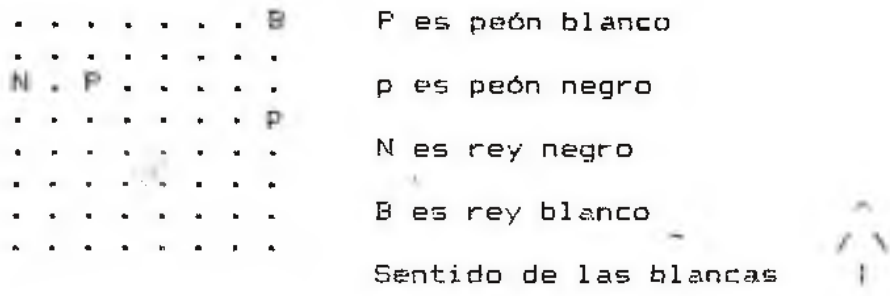
H3 => not H

Siendo R1= (X,BGX1,HGX1,MCX1) y R2=(X,BGX2,HGX2,MCX2) se define

R1 or R2 = (X,BGX1 or BGX2 , HGX1 or HGX2 , MCX1 or MCX2)

El operador 'or' en los 'goals tiene la interpretación común de una disyunción lógica. Sin embargo en cuanto a los 'move contraints' MCX1 or MCX2 se entiende como: Selecciónense movimientos que satisfagan a MCX1 o MCX2 o ambos. Se pide también un orden de movimientos y es : primero los movimientos que satisfacen tanto a MCX1 como a MCX2 posteriormente a cada uno de ellos.

De lo anterior observamos que se tiene un 'move constraints' más amplio y esto ajedrecísticamente da oportunidad de especular con la realización de alguno de los dos planes. El siguiente ejemplo en el cual juegan las blancas aclarará lo antes dicho.



El bando blanco solo puede aspirar a empatar la partida, pero se encuentra con dificultades como la de no poder defender el peón con su rey en dos movimientos, además de que no puede detener el peón negro por no estar el rey dentro del cuadrado del peón.

Esta posición nos sugiere dos planes para el bando blanco:

Plan A: coronar el peón por una dama

BG coronar peón

HG no perder peón

MC movimientos de peón y de rey en sentido horizontal

Plan B: capturar el peón negro para evitar que corone

BG capturar peón negro

HG ninguna

MC movimientos de rey en sentido vertical y diagonal

Se observa que ambos planes son refutados, el primero con la captura del peón blanco por parte del rey negro en dos movimientos, y el segundo porque el rey blanco no logra estar dentro del cuadrado del peón negro.

El el metaplan 'or' será:

BG coronar peón o capturar peón negro

HG no perder peón

MC movimientos de peón y de rey en sentido horizontal o movimientos de rey en sentido vertical y diagonal

Teniendo ahora dos metas a escoger y un 'move constraint' mayor se puede podrá empezar la partida especulando con la realización de una de las dos metas con un movimiento de rey en sentido diagonal. Esto logra ubicar al rey en un punto intermedio para lograr algún objetivo.

Metaplan Mod

Supongamos que se tiene la siguiente situación:

La hipótesis H1: Suc(P,pos)

Si P tiene éxito entonces se desprende algún resultado importante que representamos por la hipótesis H2.

Al investigar H1 se halló un plan R que refuta al plan P. Así pues, no se puede avanzar en la investigación de H2. Pero si se corrige al plan P quizá se pueda tener éxito y por tanto se probaría H2. Esta idea queda sintetizada a continuación:

Precondición

H0 : Ref (R,P,pos)

H1 : Suc(P,pos)

H1 -> H2

H0 -> not H1

Acción

Hn1: Suc (P mod P1,pos)

Hn1 -> H2

El operador mod entre planes tiene la siguiente función:
Siendo P= (X,BGX1,HGX1,MCX1) y P1= (X,BGX2,HGX2,MCX2)
entonces P mod P1 = (X,BGX1,HGX1,MCX1 mod MCX2) donde:
MCX1 mod MCX2 se interpreta como :

Selecciónense movimientos que satisfagan a MCX1 o MCX2 o ámbos. Se pide también un orden de movimientos y es : primero los movimientos que satisfacen tanto a MCX1 como a MCX2, posteriormente a MCX1 y finalmente a MCX2.

Para explicar lo que significa corregir el plan utilizaremos un ejemplo. Supóngase la siguiente posición:

```
. r t . . . . .
p p p . . . . .
. . c . . . . .
. A D . . . . .
. . . A . . . .
. . . . . . . .
. . . . . . . .
R . . . . . . .
```

R,A,D es rey,alfil y dama blancos respectivamente

r,p,t,c es rey,peón,torre y caballo negros respctivamente

Sentido de las blancas

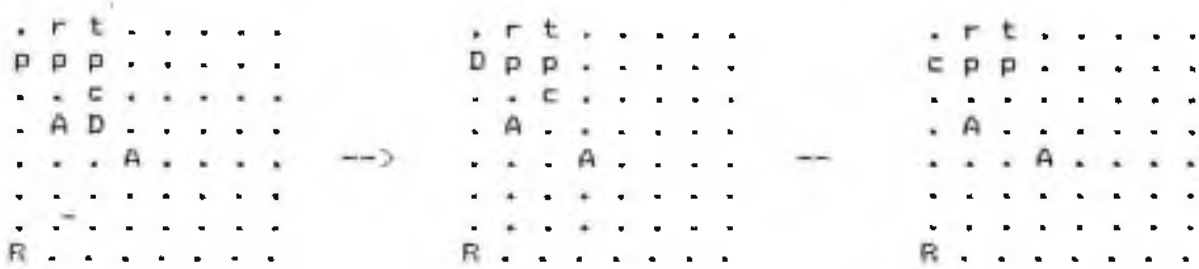
En esta posición juegan las blancas, el plan P para el bando blanco:

BG jaque mate

HG –

MC movimientos de dama

Como hipótesis H1 tendremos H1:suc(P,pos) donde P y pos es el plan y la posición de arriba respectivamente. Al investigar H1, dar mate con dama capturando el peón de la columna más a la izquierda, se encuentra H0 que refuta H1. La hipótesis H0 consistirá ,como puede verse en la secuencia de diagramas siguientes, de la captura de la dama por parte del caballo negro.



Consideremos el plan P1 obtenido de alguna manera:

BG elimina defensor

HG –

MC movimiento de alfil

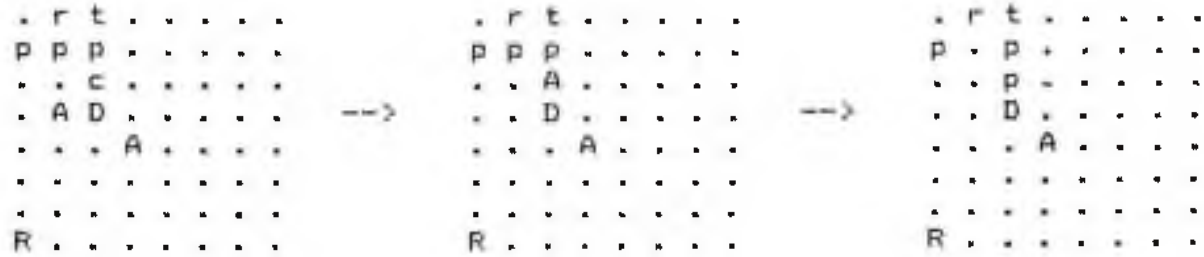
Y ahora apliquemos el metaplan mod a P y P1 y obtenemos:

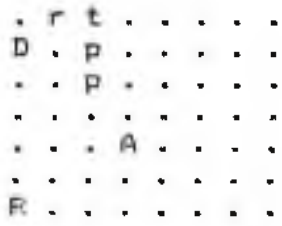
BG jaque mate

HG –

MC movimientos de dama o movimientos de alfil

Con esto se tiene el mismo BG pero con un conjunto de 'move constraint' mayor y con la posibilidad de realizar el BG. Los diagramas muetran la acción de este último lan.


. r t . . . . . .
p p p . . . . . .
. . c . . . . . .
. A D . . . . . .
. . . A . . . . .
. . . .
. . . .
R . . . .
-->
. r t . . . . . .
p p p . . . . . .
. . A . . . . . .
. . D . . . . . .
. . . A . . . . .
. . . .
. . . .
R . . . .
-->
. r t . . . . . .
p . p . . . . . .
. . p . . . . . .
. . D . . . . . .
. . . A . . . . .
. . . .
. . . .
R . . . .


. r t . . . . . .
D . p . . . . . .
. . p . . . . . .
. . . .
. . . A . . . . .
. . . .
R . . . .

3.2 Descripción del sistema y su uso

3.2.1 Arquitectura de AL3.

A continuación se describen los módulos de AL3.

Nivel general de operació de AL3.

A continuación se presenta la parte más general de la operación de AL3 en PROLOG . La claúsula resuelve efectúa los ciclos de ejecución recurriendo a la base de conocimientos por medio de la claúsula recurre_a_base .

```
problema:-
  inicialización(Objetivo,Recursos),
  resuelve(Objetivo,Recursos,Respuesta,Explicación),
  reporta(Objetivo,Respuesta,Explicación).

resuelve(Objetivo,_,Si,Nil):-
  cierto(Objetivo).

resuelve(Objetivo,_,No,Nil):-
  falso(Objetivo).

resuelve(Objetivo,Recursos,indeterminado,nil):-
  excedido(Recursos,Objetivo)

resuelve(Objetivo,Recursos,Respuesta,[Nota|Exp1]):-
  recurre_a_base(Objetivos,Recursos,Objetivo1,Recurso1,Nota),
  solve(Objetivo1,Recurso1,Respuesta,Exp1).
recurre_a_base(T,Res,T1,Res1,Note):-
  selecciona_metodo(T,Nombre_metodo,Contexto),
  ejecuta(Nombre_metodo,Contexto,Hechos,Nota,Recur_utilizados),
  actualiza(T,Hechos,T1),
  resta(Res,Recur_utilizados,Res1).
```

Traductor de código fuente a código intermedio

El sistema traduce inicialmente el código fuente a un código intermedio más adecuado para su uso. Este código fuente es la base de conocimientos que se presenta al usuario como un lenguaje declarativo mediante el cual se definen. las estrategias de solución del problema concreto. La traducción principal es transformar las funciones lógicas a su forma clausular.
Es también importante tener la información clasificada, por ejemplo distinguir hechos de hipótesis.

Máquina inferencial.

Demostrador de proposiciones.

El objetivo del sistema es hallar una prueba de la forma:

M1(x),M2(x),...,Mn(x) |- P(p) o bien

M1(x),M2,(x)...,Mn(x) |- no P(p)

donde M1,M2,...,Mn y P son fórmulas bien formadas del cálculo de predicados . Se restringen las fórmulas de manera que no contengan ningún cuantificador y un sólo término que es precisamente x. Esta x corresponde al conjunto de posiciones o estados legales del problema dado. El conjunto { M1,M2,..,Mn } corresponde precisamente a los métodos de la base de conocimientos y P a la pregunta inicial. Esta pregunta inicial se hace sobre una constante p que corresponde a la posición o situación dada del problema. Para realizar cualquier prueba, x se unifica con p, por tanto realmente se trata de una prueba del cálculo de proposiciones de la forma :

M1,M2,...,Mn |- P o bien

M1,M2,...,Mn |- no P

Así pues, en lo sucesivo nos referiremos a proposiciones. Cada Mi a su vez, puede verse como una prueba de la forma:

H1,H2,...,Hn |- M

donde cada Hi es una proposición atómica o una proposición atómica negada. Estas proposiciones corresponden a los hechos de la precondición del método. Las hipótesis de la precondición son un subconjunto del conjunto de las proposiciones de las que consiste M.

Algunas de las proposiciones atómicas corresponden a los lemas o planes. Así pues, para éstas hay una interpretación o forma particular de hallar su valor de verdad.

Operación del demostrador

A continuación se proporciona un algoritmo de operación del demostrador. Este debe considerarse cuando se efectue la corrida del sistema.

El mecanismo del demostrador puede expresarse en términos generales con el siguiente algoritmo:

```
Demo (P,M) /* pregunta P, y metodos M */
A := P ; B:= not P;
mientras M no sea vacio  efectúa proceso prueba
inicio proceso prueba
  escoge un metodo interesante (Mi,M);
  A :=  Mi -> A;
  B :=  Mi -> B;
  si teorema(A) entonces salida(A es cierto);
  si teorema(B) entonces salida(A es falso);
  M := elimina_elemento(M,Mi);
fin proceso prueba
salida(A es indecidible);
```

La función lógica corriente de A se llama objetivo positivo.
La función lógica corriente de B se llama objetivo negativo.
Estas actualizaciones corrientes se justifican por el teorema de la deducción.
Como es más fácil realizar operaciones 'or' que -> (implica) estando las funciones en forma clausular, entonces en vez de realizar

A:= Mi -> A; se efectúa A:= not Mi or A;

Los métodos Mi,se tienen entonces negados desde el principio (tarea que corresponde realizar al traductor de codigo fuente a intermedio)

3.2.2 Fundamentos teóricos del sistema

Manejo de las funciones lógicas

Cada función lógica se puede representar como un conjunto de claúsulas (que llamaremos líneas), cada una de la forma :

a1,a2,...,am => b1,b2,...,bn significando

a1 & a2 & ... & am => b1 + b2 + ... + bn

donde & es el conectivo 'and' y + es el conectivo 'or' .

Utilicemos la notación de conjuntos de la siguiente manera: Letras mayúsculas A,B,... denotarán conjuntos de proposiciones atómicas.

Si A= {a1,a2,...,am} y B={b1,b2,...,bn} entonces

A -> B representa la linea a1,a2,...,am -> b1,b2,...,bn

En esta notación una función lógica puede escribirse como:

| A1 -> B1 |
| A2 -> B2 |
| .... |
| .... |

Si Ai = vacío entonces representa el valor de verdad 'cierto'. Si Bi = vacío entonces representa el valor de verdad 'falso'.

Una línea es una tautología si

(1) A es falso, o

(2) B es cierto o

(3) A intersección B <> vacío

Un objetivo es un teorema si todas sus líneas son tautologías.

Dados dos objetivos T1 y T2, la composición lógica or entre estos (también llamado producto de objetivos) se obtiene de la siguiente manera:

```
                ¦ A1 -> B1 ¦                  ¦ C1 -> D1 ¦
Sea T1 =        ¦ ........ ¦   y    T2 =      ¦ ........ ¦
                ¦ Am -> Bm ¦                  ¦ Cn -> Dn ¦
```

entonces el objetivo T=T1 or T2 , también denotado T= T1 x T2 es el conjunto de líneas X -> Y tal que

X = Ai unic Cj

Y = Bi unión Dj para todo i en [1,m] y j en [1,n]

Actualización de objetivos.

El objetivo se actualiza por medio de productos cuando se utilizan nuevos métodos o simplemente insertando el valor de verdad de alguna hipótesis, cuyo valor sea descubierto por medio de lemas, planes o refutaciones.

Operaciones que simplifican los objetivos

(1) Borrando líneas que sean tautologías.

(2) Borrando líneas que sean implicadas por otras líneas.

Una línea A -> B implica otra línea A1 -> B1 en un objetivo si A es subconjunto de A1 y

B es subconjunto de B1

(3) Insertando valores de verdad a hipótesis cuyos valores de verdad sean implicados del objetivo.

Un valor de verdad de una hipótesis h es implicado si h aparece en el mismo lado de todas las líneas de un objetivo positivo y en este mismo lado en todas las líneas del objetivo negativo. El valor es:

(a) Cierto si aparece del lado izquierdo

(b) Falso si aparece del lado derecho.

Se supone consistencia, es decir que no es posible que tanto el objetivo positivo como el negativo sean teoremas.

Método interesante.

Un método es interesante si se satisface su precondición y si ésta incluye la hipótesis más interesante. Un método es más interesante que otro si es más económico que otro.

Hipótesis más interesante.

La hipótesis más interesante es la que tiene más ocurrencias en el objetivo positivo y negativo.

3.2.3 Ejemplo en AL3

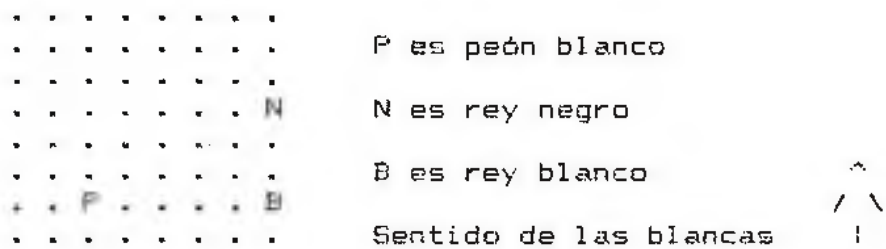
Jugadores de alto nivel y jugadores no tan fuertes pueden similar una partida en la apertura y medio juego, sin embargo los los primeros tienden a imponerse en el final.

Los finales de juego en ajedrez requieren de mucha estrategia para ganar. Es necesario conocer varios planes para llevar a cabo una partida en forma correcta.

El final de juego que se discutirá a continuación corresponde al final de rey y peón contra rey.

Este ejemplo está relacionado con la base de conocimientos tratada en la sección 3.1.3. La explicación detallará algunos enunciados de la base de conocimientos.

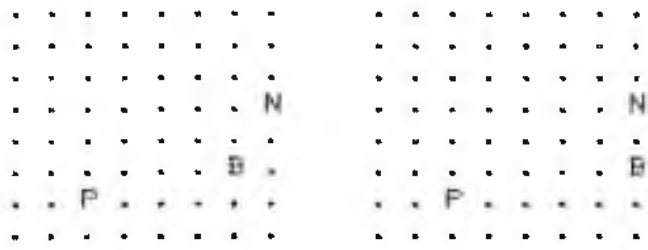
Aunque se trata de uno de los finales de juego más simples no es tan trivial para jugadores sin experiencia. El siguiente ejemplo demostrará lo dicho.



Siendo el turno del blanco qué deberá jugar ?

Puede ganar el blanco ?

En caso afirmativo ,tiene un movimiento único para ganar o son varios ?



Si el blanco juega como en (a) la partida debe finalizar tablas, sin embargo si juega como en (b) deberá ganar y la jugada es única.

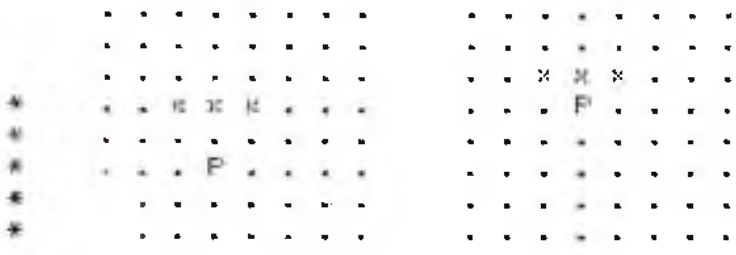
Dos ideas centrales para ganar un final de peón y rey contra rey son :

- Uso de la regla del cuadrado

- Ocupación con el rey blanco de ciertas casillas críticas

La regla del cuadrado ya se mencionó previamente, cuando se abordó el punto de la importancia del conocimiento en ajedrez en la introducción.

Definición de las casillas críticas:



(a) (b)

Figura 1

Las casillas críticas se denotan con la letra X. Se tienen 2 casos:

- Cuando el peón se halla en cualquiera de los primeros renglones como en la figura 1a .

- Cuando el peón se halla sobre el quinto renglón como en la figura 1b.

Para los lectores interesados se les refiere a [I. Mazielis, 3]. En lugar de decir que un peón se encuentra en el renglón n diremos que el rango del peón es n.

En cada caso, si el rey blanco puede ocupar estas casillas críticas entonces debe ganar. Claro est., se supone que el rey negro no pueda capturar al peón. Para el peón de torre sólo se cuenta con una casilla crítica como se muestra en la figura 2.

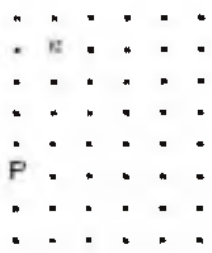


figura 2

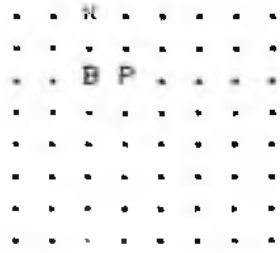


figura 3 rey negro no en x

Esta casilla crítica es independiente del sitio del peón de torre de la primera columna. Simétricamente se tiene una casilla crítica para el peón de torre de la última columna.

Estos dos principios, la regla del cuadrado y las casillas críticas proveen un conocimiento básico para definir una estrategia de juego para este final.

Definimos los siguientes patrones de posiciones para utilizarlos en la estrategia.

Nombre de patrón	Descripción
corre_peón	Cuando el peón puede coronar de acuerdo a la regla del cuadrado
peón_torre	Cuando el peón es de torre
peón_r7	Cuando el rango del peón es 7
patrón_bp6	Patrón de la figura 3
casillas_críticas	Si el rey blanco se halla en las casillas críticas
n_a_salvo	Si el peón no puede ser capturado en el movimiento corriente

Con estos patrones podemos definir las siguientes reglas:

- Si peón_corre entonces blanco gana
- Si patrón_bpó entonces blanco gana
- Si casillas_críticas entonces blanco gana
- Si el rango del peón es 7, el blanco gana si puede coronar
- Si puede obtenerse peón_corre entonces blanco gana
- Si puede obtenerse patrón_bpó entonces blanco gana
- Si puede obtenerse casillas_críticas entonces el blanco gana

En las últimas 3 reglas el 'Si puede obtenerse debe enunciarse en términos de planes. Se nota inmediatamente que las reglas enunciadas sirven para averiguar si el blanco gana. Conviene agregar reglas pensando del lado del bando negro para intentar empatar la partida. Para esto se definen los siguientes patrones:

Nombre del patrón	Descripción del patrón
Pierde_peón	Si en el siguiente movimiento legal del negro es posible capturar al peón
peón_bloqueado	Cuando el peón se halla bloqueado de acuerdo a las figura 2a

figura 2a

En esta figura lo que importa es que el peón esté en un rango menor que 7 y rey negro en x.

Se proponen las siguientes reglas para el negro :

- Si pierde_peón entonces empate
- Si peón_bloqueado entonces empate
- Si puede obtenerse pierde_peón entonces empate
- Si puede obtenerse peón_bloqueado entonces empate
- Si puede obtenerse ahogado entonces empate

Nuevamente las últimas 3 reglas deben expresarse en términos de planes, pero ahora para el negro.

Debe incluirse una regla que exprese la relación lógica ent e empate y ganan blancas, es decir

- empate si y solo si no ganan blancas

Sigue ahora proponer los planes. Es característico en estos finales los movimientos de reyes, principalmente buscando la oposición entre estos, esto es, cuando los reyes se encuentran en una misma fila, columna o diagonal [I. Mazielis, 6].

Acercar los reyes al peón manteniendo ciertas condiciones puede considerarse una buena heurística, con ciertas reservas.

Una base de conocimientos que expresa algunas de las ideas anteriores es precisamente la presentada en el capítulo anterior, asi que a continuación se mostrará el funcionamiento de ésta en algunos ejemplos con el propósito principal de ver el funcionamiento de ALS.

También se espera que quede más o menos claro el procedimiento seguido en el diseño de estas bases.

Para el ejemplo siguientes guíese con el punto referente a Operación del demostrador dentro de la sección 3.2 para tener una idea más clara del funcionamiento del sistema.

Ejemplo

Entrada al sistema:

```
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. B . . N . . .)
(. . . . . .)
(. . . P . . . .)
(. . . . . .)
(. . . . . .)
```

PREGUNTA: GANA

Inicio del procedimiento :

"INICIO DE RESOLUCION "

"OBSERVA " ((PEON_T 0))

Lo primero que siempre hace es sistema es llevar a cabo las observaciones de la base de conocimientos. En este caso, correspondió a determinar que PEON_T es falso (denotado con 0) es decir que no se trata de peón de torre.Lo que sigue es la secuencia de ciclos de ejecución.

"HIPOTESIS INTERESANTE " GANA

En el primer ciclo de ejecución la hipótesis interesante corresponde a la pregunta del usuario.

" ELECCION " (METODO M0)

Se cuentan con los métodos M0 y M3 para investigar a GANA, considerando que el peón no es de torre. Se escogió M0 arbitrariamente pues ámbos métodos tienen la categoría de FACIL.

"HIPOTESIS INTERESANTE " GANA_PEON_NT

" ELECCION " (METODO M2)

"HIPOTESIS INTERESANTE " CORRE_PEON

" ELECCION " (LEMA CORRE_PEON)

"HIPOTESIS INTERESANTE " CASILLAS_CRITICAS

" ELECCION " (LEMA CASILLAS_CRITICAS)

"HIPOTESIS INTERESANTE " PATRON_BP6

Cuando como en esta ocasión el sistema despliega más de una vez el letrero HIPOTESIS INTERESANTE en forma consecutiva quiere decir que el sistema realizó una primera elección errónea que tuvo que corregir. Esta propiedad la hereda AL3 de PROLOG en lo que se llama 'backtracking'.

"HIPOTESIS INTERESANTE " GANA

" ELECCION " (METODO M3)

"HIPOTESIS INTERESANTE " TABLAS

" ELECCION " (METODO M4)

"HIPOTESIS INTERESANTE " PIERDE_PEON

" ELECCION " (LEMA PIERDE_PEON)

"HIPOTESIS INTERESANTE " PEON_BLOQUEADO

" ELECCION " (LEMA PEON_BLOQUEADO)

"HIPOTESIS INTERESANTE " PATRON_BP6

"HIPOTESIS INTERESANTE " GANA

"HIPOTESIS INTERESANTE " GANA_PEON_NT

" ELECCION " (METODO M5)

"HIPOTESIS INTERESANTE " (SUC P_BLANCO_NT)

Se escoge un plan por primera ocasión.

" ELECCION " (METODO M7)

"HIPOTESIS INTERESANTE " (REFUTA P_NEGRO P_BLANCO_NT)

" ELECCION " (REFUTA P_NEGRO P_BLANCO_NT)

Conviene checar las refutaciones a un plan primero. Es menos costoso.

```
"REFUTA RESULTO " 0
"HIPOTESIS INTERESANTE " PATRON_BP6
"HIPOTESIS INTERESANTE " GANA
"HIPOTESIS INTERESANTE " GANA_PEON_NT
"HIPOTESIS INTERESANTE " TABLAS
"  ELECCION " (METODO M9)
"HIPOTESIS INTERESANTE " (SUC P_NEGRO)
"  ELECCION " (PLAN P_NEGRO)
"PLAN RESULTO " 0
"HIPOTESIS INTERESANTE " (SUC P_BLANCO_NT)
"  ELECCION " (PLAN P_BLANCO_NT)
"PLAN RESULTO " 1
```

Sal da del sistema:

RESPUESTA (SI)

JUSTIFICACION

TABLAS RESULTO FALSO

GANA_PEON_NT RESULTO CIERTO

GANA RESULTO CIERTO

(SUC P_BLANCO_NT) RESULTO CIERTO

(SUC P_NEGRO) RESULTO FALSO

(REFUTA P_NEGRO P_BLANCO_NT) RESULTO FALSO

PEON_BLOQUEADO RESULTO FALSO

PIERDE_PEON RESULTO FALSO

CASILLAS_CRITICAS RESULTO FALSO

CORRE_PEON RESULTO FALSO

PEON_T RESULTO FALSO

Puede notarse que la justificación es muy primitiva,pues tan solo reporta los hechos obtenidos.

3.2.4 Evaluación y eficiencia de AL3.

El sistema desarrollado para esta tesis es ineficiente tanto en tiempo de ejecución como en uso de memoria. Esto se comprende considerando que sea trata de un intérprete de alto nivel que se apoya sobre otro intérprete también de alto nivel (Micro-Prolog). Sin embargo el hecho de que esté implementado en Prolog es sustancialmente útil debido a que muchos conceptos de AL se apoyan en la lógica y por otro lado se pueden realizar modificaciones imples en programación con una repercusión fuerte en el funcionamiento del sistema.

be quedar claro que lo fundamental es incursionar en este campo la 'programacion lógica' y en el área de los resolvedores tomáticos de problemas.

Sin embargo, tomando en cuenta los procedimientos que se utilizan, es bastante claro que éstos en si mismos, no son complejos en tiempo o espacio. Así pues, es posible tener una implementación satisfactoria en algún otro lenguaje (por ejemplo en C) , pero es recomendable esperar a que las investigaciones en maduren aún más . Esto no quiere decir que AL3 esté en un el de inmadurez o incertidumbre, sino que se presenta tan interesante y rico en posibilidades que se conjetura que existe cho por hacer todavía, como se mencionará en el siguiente pítulo.

3.2.5 Conclusiones sobre AL3 y posibilidades de desarrollo.

AL3 se presenta como una de las formas más interesantes para formalizar la teoría de juego en ajedrez. Los conceptos que utiliza tales como métodos, planes y refutaciones, resultan muy característicos en la estrategia moderna del ajedrez. Una base de conocimientos en AL3 parece aventajar a cualquier libro de ajedrez desde un punto de vista formal . Un lenguaje de esta naturaleza presenta las siguientes posibilidades:

(1) Puede ser entendido por cualquier ajedrecista, dada la naturalidad de sus conceptos en este campo.

(2) Puede ser entendido por cualquiera que tenga cierta formación matemática, dado que el formalismo que utiliza tiene un estilo matemático.

(3) Puede ser interpretado por una computadora, teniendo un intérprete de AL3, lo que permite experimentar y corroborar la teoría.

En cuanto a sus posibilidades de desarrollo se encuentran dos alternativas interesantes y no del todo disjuntas:

(1) Desarrollarlo hacia un resolvedor general de problemas.

(2) Orientarlo aún más hacia el ajedrez.

A continuación se discute brevemente cada posibilidad.

AL3 como un resolvedor general de problemas

Aunque algunos de los conceptos de alto nivel que utiliza tienen un sentido natural en juegos con estrategia como el ajedrez, AL3 puede ser visto como un resolvedor general de problemas que utiliza un formalismo especial para la representación de problemas.

Sin embargo, puede ser un tanto forzado utilizarlos en otro contexto. Esto es debido quizá a que la definición de satisfactibilidad de un consejo depende de hallar un 'forcing tree'. Este último concepto se encuentra definido con una aparente dirección a la teoría de juegos. La simple redefinición del 'forcing-tree' alteraría todo el sistema, probablemente dando origen a AL4, pues éste se encuentra apoyado en la noción del 'consejo'.

AL3 orientado hacia el ajedrez.

En el estado actual AL3 se muestra lo suficientemente maduro para explotarlo en ajedrez. Sin embargo se podría intentar hacer un poco más sobre todo en las siguientes formas:

(1) Búsqueda de alguna base universal de predicados en ajedrez' o al menos 'predicados' importantes para las distintas etapas del juego (apertura, medio juego y final) de manera que se tengan a la disposición como biblioteca y que estén incluso implementados en algún lenguaje de bajo nivel.

(2) Trabajar en la eficiencia del sistema. En este punto conviene estudiar las estrategias para la representación de objetivos, para saber en qué casos realizar productos y cuándo puede evitarse esto, representando entonces al objetivo como un producto de subobjetivos. También es interesante estudiar la elección de las hipótesis interesantes y la búsqueda de los métodos más económicos.

(3) Explorar más alternativas en la metabase de conocimientos según lo indica Bratko. En términos dialécticos pudieran considerarse al plan , refutación y metaplan como la

tesis , antítesis y síntesis respectivamente. Pero en el estado actual la síntesis se halla en un estado bastante primitivo.

(4) Un sistema evolutivo de planes, en base al uso y desuso de éstos. La metabase de conocimientos sobre planes permite generar nuevos planes a partir de otros. Habría que observar cuáles tienden a ser más útiles para incorporarlos a la base de conocimientos para poder utilizarlos cuando sean necesario en vez de estarlos generando en cada ocasión. Similarmente habría que deshacernos de aquellos que parezcan inútiles con el tiempo.

(5) Introducir más conceptos del ajedrez, tales como profilaxis, sobreprotección, etc. Aún cuando estos conceptos puedan definirse en términos de los elementos actuales de AL3, sería mejor que se les otorgara un tratamiento especial. Y es que la sobreprotección en ajedrez trasciende a la estrategia pasando a ser un 'valor' en el ajedrez. Esto sugiere que sería interesante el poder definir precisamente valores en AL3. Entendiendo a éstos como aspectos que deben cuidarse siempre (o casi siempre) independientemente de la estrategia a seguir. O quizá, siendo más flexible, se puedan tener ciertos valores para familias de estrategias.

(6) Intentando observar a AL3 como un ajedrecista humano, podría catalogársele del tipo 'científico' y que siempre sabe 'mas o menos ' hacia donde va. Esto es debido a su sistema de métodos, planes y refutaciones. La metabase de conocimientos es la que podría otorgarle cierta 'creatividad'. Sin embargo, sería conveniente que en ocasiones explorara o experimentara para enriquecer su experiencia y posibilidades de juego.

4. Aprendizaje

El presente capítulo tiene como intención principal discutir algunas técnicas de aprendizaje principalmente orientadas a juegos y el uso de algunas de ellas en los lenguajes de consejos. Me parece sin embargo necesario iniciar esta exposición con una introducción general sobre este tema , la cual iniciará con una definición tentativa de aprendizaje. Esto me parece importante por cuestión de orden a pesar de que para efectos prácticas pueda ser un tanto inecesario.

4.1 Introducción

Partamos de la siguiente definición de aprendizaje para que de una u otra forma supongamos estar centrados en el tema:
"Entendemos por aprendizaje de un sistema a cualquier cambio adaptativo de éste que le permita, al repetir una tarea sobre la misma población realizarla más efectiva y eficientemente".
Tomamos esta definición solo como un punto de partida,sin aceptar en ningún momento que sea una definición completa o siquiera correcta. Ni siquiera queda claro que se entiende por 'efectiva' y 'eficientemente'. Particularizando el término 'sistema' en 'ser vivo' ocurrirá que el concepto de instinto (o conducta específica de la especie) bien podría satisfacer esta definición, y sin embargo muchos psicólogos no estarían de acuerdo [Hilgard y Bower 1984].

Aún cuando no se tenga una definición precisa de este concepto, muchas disciplinas se han preocupado por el estudio del 'aprendizaje' . Fisiólogos, bioquímicos, biofísicos y principalmente psicólogos tienen intereses legítimos en ella.

Esto es fácil de entender considerando que gran parte de la múltiple conducta del hombre es producto del aprendizaje.

Más recientemente la I.A. ha dedicado un considerable esfuerzo al estudio del 'aprendizaje' . Se desea que un sistema aprenda precisamente para que repita una tarea más eficientemente. Más pretencioso sería pedir que el sistema pudiera aprender 'cosas' que nosotros (los humanos) ignoramos y que nos fueran de utilidad.

Cómo investigar al aprendizaje.

Una forma general para investigar no sólo al aprendizaje sino otras actividades mentales superiores es observar y estudiar al humano realizando estas funciones.

Para esto se usa una técnica llamada 'pensamiento en voz alta'. Consiste en tomar grupos de individuos y pedirles que expresen sus pensamientos en voz alta, a la vez que realizan alguna tarea intelectual. Todo pensamiento, por intrascendente que le parezca al sujeto debe verbalizarlo. Se graban todas la entrevistas y posteriormente se analizan, buscando similitudes, diferencias y toda información que pueda ser útil. Ocurre que se encuentran ciertos indicios interesantes y se puede volver a entrevistar al grupo para reforzar o desechar alguna sospecha.Cuando la investigación llega a su estado de maduración se puede entonces intentar construir la teoría propuesta por medio de algún lenguaje computacional que pueda ser interpretado por alguna máquina.

Algunas características que se han encontrado primordiales en la modelación del aprendizaje.

a) Se requieren principalmente procedimientos heurísticos en vez de algorítmicos.

Al respecto basta con mencionar lo que Arthur L. Samuel (creador de un famoso programa que juega damas y aprende) indica:

'Concluimos que en nuestro estado actual de conocimientos, el único enfoque práctico, aun con la ayuda de la computadora digital, será mediante el desrrollo de la heurística que tienda a copiar el comportamiento humano'[Samuel 1967].

b) Reconocimiento de patrones.

Regular o irregular, la alternancia del día y la noche fue probablemente el primer 'patrón' de tiempo que impresionó al hombre. También hubo de existir un reconocimiento temprano de los estereotipados gritos de los animales previendo la tormenta, expresando el desafío y la derrota.

Patrón se identifica con orden, organización. Un primer atributo característico del patrón es poderlo memorizar y compararlo con otro. En contraposición está el concepto 'casual'. Cuando no podemos identificar algo con regularidad, o sea establecerlo dentro de algún patrón, entonces este 'algo' es casual. Pero cuando estas casualidades empiezan a tener cualidades en común, se nos empiezan a parecer familiares, hasta el punto de que formamos un patrón de todas éstas. En este momento ya no son casualidades. EL intelecto se reconforta en hallar patrones.

En el aprendizaje es fundamental el reconocimiento de patrones. Tal es el caso, que podría identificarse el aprendizaje

precisamente con el 'descubrimiento de patrones' y su incorporación en la memoria para uso posterior.

En el ajedrez esto parece ser muy claro. Los grandes maestros del ajedrez han adquirido a través de su entrenamiento una cierta cantidad de patrones posicionales, con estrategias asociadas. Estrategias que también se presentan como patrones de alternativas a seguir. Un excelente ajedrecista puede dirigir una partida con un buen nivel de juego en escasos minutos. Dado que no tiene que calcular movimientos sino tan solo poner a funcionar su maquinaria de reconocimiento de patrones.

Es por esto que los modelos computacionales del aprendizaje no pueden pasar por alto este aspecto primordial. Inclusive, algunos de los modelos computacionales sobre aprendizaje están basados exclusivamente en el reconocimiento de patrones. Las técnicas matemáticas basadas en estádistica y/o en lenguajes formales son los principales recursos que se usan para el manejo de patrones.

Un ejemplo de programa que descubre patrones es BACON .4 . Este programa descubre leyes sobre regularidades de cuerpos expresadas por medio de datos numéricos. Por ejemplo, BACON .4 redescubrió rápidamente una de las leyes de Kepler.

Aquí, el concepto de descubrimiento se identifica con aprendizaje. Pues cuando se descubre algo se puede utilizar para actuar en forma más acertada respecto a lo descubierto.

c) Razonamiento inductivo.

La inducción de reglas por medio de muchos ejemplos es una forma de aprendizaje. Estrictamente hablando las reglas son precisamente patrones hallados. La distinción puede ser en un momento convencional, según se formalicen los conceptos.

Razonamiento deductivo.

El razonamiento del cálculo de predicados parece ser muy importante para llevar a cabo ciertas tareas que incluye el proceso de aprendizaje. Sin embargo, los humanos proceden en general en forma más 'impredecible' que en la lógica formal. Es por esto que también es útil el razonamiento bajo 'incertidumbre' basado principalmente en la teoría de las probabilidades. Como un ejemplo simple, podemos citar un programa de aprendizaje de conceptos, realizado con estas características. Al empezar se dota al modelo de una larga lista de hipótesis relativas a la forma de la solución. Tales hipótesis tienen diversos pesos de probabilidades que determinan su selección . Se selecciona y coteja una hipótesis con los datos; si es válida en las instancias pasadas, se utiliza hasta confirmarla. Cuando no se confirma, su peso de probabilidad disminuye, y empieza de nuevo la selección de hipótesis .

Este es un ejemplo muy simple de generación o ajuste de reglas probabilísticas.

Métodos de aprendizaje.

Suelen separarse los métodos de aprendizaje según la clasificación siguiente o alguna otra. La verdad es que en muchos casos se halla una mezcla de estos métodos donde incluso es difícil señalar donde empieza uno y termina otro.

a) Por implantación

El sistema no requiere realizar ninguna inferencia o tipo de transformación sobre el conocimiento que recibe. Lo anexa tal como viene.

b) Por instrucción

El sistema desarrolla alguna inferencia, a partir de la cual obtiene un nuevo conocimiento; sin embargo, la tarea de definir el uso de este conocimiento a partir de un mayor número de inferencias corre a cargo del sistema.

c) Por analogía

El sistema busca similitudes en el problema a resolver con un conjunto de problemas que tiene dados con sus respectivas soluciones. Cuando halla un problema muy similar, intenta resolver el problema en forma similar, con posibles ajustes.

d) Por medio de ejemplos

Dado un conjunto de ejemplos y contraejemplos de un concepto, el sistema infiere una descripción general que describa a los ejemplos y excluya a los contraejemplos. Es quizá el caso más explorado actualmente en la I.A.

e) Por observación y descubrimiento

Al observar un grupo de diferentes patrones dados, el sistema deberá descubrir cuáles son sus similitudes y cuáles sus diferencias, con el propósito de encontrar reglas que definan al conjunto al que pertenece cada patrón observado.

4.2. Algunos modelos de aprendizaje

A continuación se mencionarán algunos modelos de aprendizaje propuestos para juegos.

- **Modelo de Michie para jugar gatos.** [M.A. Murray-Lasso 1985]

Este modelo es muy interesante por su simplicidad y esta basado en la idea de recompenza y castigo. El modelo se basa en una máquina construida de cajas de cerillos y chochitos de colores. Expliquemos primero como se juega con esta máquina y posteriormente como se aprende.

- **Módulo de juego.** La máquina consiste de una colección de cajas de cerillos, cada una de las cuales corresponde a una posición del juego (combinación de cruces, círculos y espacios vacios). Podemos utilizar simetrías y reflexiones para reducir el número de cajas requeridas. Dentro de cada caja de cerillos hay un conjunto de cerillos de diferentes colores, cada uno de los cuales representa una movimiento legal en esa posición. Así por ejemplo, el rojo puede representar la jugada en que una cruz se coloca en la posición 1,1 del gato. Cada caja tiene un agujero por el cual puede salir uno de los chochitos, el cual se selecciona al azar agitando previamente la caja antes de sacarlo. Supongamos que la máquina comienza teniendo en todas la cajas de cerillos chochitos de colores en iguales proporciones. El juego consiste en seleccionar movimientos (como se indicó anteriorménte) según la caja correspondiente a la posición corriente del juego.

- **Aprendizaje.** Al final de un juego, cuando se sabe si la máquina gano perdió o empató, es el momento del aprendizaje por medio de refuerzos positivos o negativos, el cual se logra

cambiando las cantidades de chochitos en las cajas que se tomaron durante el juego. Spongamos por ejemplo que la máquina perdió. Entonces en la última caja escogida se eliminan todos los chochitos del color que se seleccionó. De esta manera, la siguiente vez que se presente la misma situación la máquina no realizará el movimiento con el que perdió inmediatamente. También conviene penalizar a los chochitos escogidos en la penúltima caja. Probablemente no debamos eliminar todos los chochitos del color seleccionado, sino tan solo reducirlos, digamos a la mitad. Pero incluso también conviene penalizar a los chochitos de la antepnúltima caja. Sin embargo, como la evidencia es circunstancial, la penalización debe ser ligera, por lo que podrían reducirse los chochitos del color seleccionado en un 25%. Con esto, la siguiente vez que se juege, la máquina tendrá menos tendencia a efectuar ciertos movimientos que de una forma u otra condujeron a la derrota. Por otra parte es posible que la máquina gane un juego. En estos casos, habrá que reforzar los chochitos elegidos aumentándolos. En la caja correspondiente a la última jugada, los podríamos reforzar infinitamente a base de poner un número muy grande de chochitos del color correspondiente al movimiento ganador, o lo que es equivalente, eliminando todos los chochitos de los colores restantes. En las jugadas anteriores al final, el refuerzo no deberá ser tan fuerte. Digamos que en la inmediata anterior, dupliquemos los chochitos que salieron y en la anterior a ésta los aumentemos en un 25%. El profesor Michie jugó un torneo contra esta máquina y después de un cierto número de partidas la máquina aprendió, resultando una "experta jugadora

de gato".

- Aprendiendo por refuerzos en los lenguajes de consejos.

Tanto A11 como A13 pueden utilizar estas ideas para aprender. Sin embargo discutiremos tan solo el caso para A13. Recordemos que en A13 existen métodos para resolver problemas. Se mencionó que dichos métodos pueden tener pesos que indicaban la dificultad de su ejecución. Pero también es cierto que los métodos pueden fracazar en su misión o tener éxito. Aquí es donde podemos tener un recurso para recompenzar o castigar los métodos. De esta manera podemos evantualmente encontrar una relación de orden adecuado entre estos para escogerlos.

- Modelo de Samuel para jugar damas

Este modelo que se llevó a una practica real es muy interesante tanto por sus resultados obtenidos (ha vencido a un campeón estatal y empatado con otros), como por su capacidad para aprender.

- Forma del juego

Antes de discutir sus técnicas de aprendizaje conviene explicar como juega para situarnos en el contexto.

Utiliza la técnica alfa-beta ya explicada en el capítulo I, con una función de evaluación basada en un polinomio lineal. Cada término del polinomio corresponde a un criterio o parámetro de la posición. El coeficiente indica el peso o la importancia de dicho parámetro. Esto es análogo a la función de evaluación propuesta por Shannon mencionada en el cápitulo I. Aquí, sin embargo, vale la pena señalar que se hace un análisis interesante sobre la manera de incrementar la poda alfa-beta aumentando la probabilidad de que primero sean explorados los mejores senderos [Samuel 1967]. Se recomienda leer el articulo mencionado.

-Técnicas de aprendizaje

Básicamente se consideran dos procedimientos de aprendizaje:

- Aprendizaje por memoria

Expliquemos este procedimiento brevemente: Supóngase que se esta jugando con una anticipación de 3 movimientos. Podemos almacenar ciertas posiciones con su correspondiente valor asociado alfa-beta. La proxima vez que nos encontremos con alguna de estas posiciónes como hoja del árbol de exploración, en vez de calcular la función de evaluación, utilizamos el valor

registrado previamente. Con esto logramos una anticipación o profundidad de analisis de 6 movimientos en vez 3 con respecto a la variante o variantes que incluyan estas posiciones. Algoritmos especiales deben utilizarse para decidir que posiciones almacenar, como hacerlo y que tanto limitar el número de posiciones a memorizar. En [Samuel 1963] se discuten más ampliamente estas ideas. Lo que importa para esta tesis es saber como incorporar esta idea en los lenguajes de consejos.

- Aprendizaje por memoria en los lenguajes de consejos.

Se mencionó que limitamos los árboles forzados con profundidad predefinida . Supongamos que en una cierta posición hallamos que se satisface un consejo (porque hallamos un árbol forzado, tal como la definición nos lo exige). Podemos almacenar la posición junto con el consejo. La próxima vez que tengamos que checar la satisfactibilidad del consejo mediante la búsqueda del árbol forzado, podemos verificar si en una posición hallada en el 'arbol forzado se satisface el consejo, por medio de la memoria construida sobre posiciones y consejos, en vez de seguir explorando el árbol. Así, podemos aumentar la profundidad de análisis de una posición.

Esta forma de aprendizaje pudiera considerarse poco útl para el ajedrez en general. Y esto es cierto debido a la gran cantidad de posiciones que pueden existir. Sin embargo, tanto para la apertura como para cierta clase de finales, es factible el uso de esta idea.

- Aprendizaje por generalización.

Una forma obvia de decrementar la cantidad de espacio

requerida para utilizar experiencia previa, es generalizar en base a la experiencia y memorizar solo la generalización realizada. La manera de llevar esto a cabo es por medio de un ajuste de los coeficientes (pesos del parámetro) del polinomio lineal. Antes de discutir como ajustar el polinomio vale la pena preguntarnos como saber si el ajuste (se lleve a cabo como sea) corresponde a un aprendizaje. Es decir (retomando la definición dada de aprendizaje) cómo establecer que el polinomio ajustado es más eficiente que el polinomio original. Considerando eficiencia como 'jugar mejor' como saber que se juega mejor. en [Samuel 1963] discute este punto y se mencionan ciertas ideas. Para el caso del ajedrez me parece que podrían considerase dos formas. El uso del 'rating' y/o el punto de vista de un experto. En general lo se debe tener son dos versiones del programa; el original y modificado (o sea el que supuestamente aprendió). Mantener ambos durante un tiempo bajo prueba y si los resultados (rating y criterio del maestro) indican que efectivamente la nueva versión es mejor que la anterior, aceptamos el ajuste y reiniciamos el proceso de aprendizaje . Aceptando que de alguna forma podamos verificar el proceso de aprendizaje como se mencionó, pasemos a determinar cómo ajustar los coeficientes. Sea P el polinomio inicial. Supongamos (apriori) que existe un polinomio P1 (con los mismos términos, pero distintos coeficientes) tal que P1 es 'mejor' que P.' Supongamos además que aunque no sepamos explícitamente quien es P1 si sepamos algo de él. Concretamente supondremos que tenemos una muestra de posiciones y un movimiento elegido según P1. Llamemos a este movimiento, el movimiento correcto.' Técnicas de estadística nos pueden indicar como

corregir el polinomio P de manera que se aproxime a P1. En [Samuel 1963] se reporta el siguiente método: Contemos (en h) simplemente el número de la jugadas, para cada parámetro por separado para las cuales el valor paramétrico es mayor que el valor asociado con la jugada correcta y contemos (en l) el número de jugadas para las cuales el valor paramétrico es más pequeño que el valor asociado a la jugada correcta. Estos recuentos se acumulan para todas las posiciones de la muestra en H y L. Entonces una medida de bondad del parámetro para predecir la jugada correcta estará dada por C=(L-H)/(L+H). Esta fórmula tiene las dimensiones de un coeficiente de correlación; tendría un valor de +1 si el parámetro en cuestión predice siempre la jugada correcta, un valor de -1 si nunca hiciera una predicción correcta, y un valor de 0 si no hubiera correlación alguna entre las indicaciones del programa y la jugada correcta. El procedimiento escogido fue utilizar los valores de la C así obtenidos como los coeficientes en el polinomio de evaluación.

Lo que ahora debemos responder es de donde surgió la muestra de P1. Aprender de los libros, aprender de un rival bien capacitado, o incluso aprender de uno mismo haciendo un análisis más profundo, son algunas formas de donde surge P1.

- Aprendiendo de libros. En un libro uno cuenta con partidas jugadas y analizadas por expertos. Estas partidas pueden corresponder a la muestra P1 mencionada. Incluso filtramos la partida según el análisis del libro de manera que evitemos incluir movimientos erroneos que realizaron los expertos.

- Aprendiendo de un rival capacitado. Básicamente

corresponde al punto anterior, solo que este mismo rival puede hacer el papel del analista al final de la partida.

- Aprendiendo de uno mismo. Suponiendo que analizando más profundamente en el árbol se pueda tener un mejor juicio, podemos aprender de la siguiente forma: Después de jugar una partida con una cierta profundidad de analisis podemos estudiar cada posición encontrada en la partida con mucho más detenimiento (aumentando la profundidad del árbol) registrando los movimientos sugeridos de esta forma. Después operamos como en los otros casos.

Dificultades y limitaciones de este método de generalizaciones. Dos dificultades fundamentales son:

- La hipótesis del polinomio lineal. No necesariamente una evaluación debe corresponder a un polinomio lineal. En [Samuel 1967] se cita una forma más general para trabajar con funciones d evaluación.

- Poca creatividad. Sólo es posible modificar los coeficientes del polinomio. Pero sería necesario 'crear' términos del polinomio.

Como una limitación básica es que necesitamos muestras grandes cuando en ocaciones es posible aprender de una solo ejemplo.

- Cómo manejar esta forma de aprendizaje en los lenguajes de consejos.

Una manera (que en princio parece un tanto forzada) de aprovechar o utilizar esta idea en los lenguajes de consejos, surge de una posible interpretación de los términos del polinomio de evaluación. Esta interpretación [Samuel 1963] consiste en considerar que los términos con coeficientes pequeños miden criterios relacionados con metas intermedias con respecto a los criterios medidos con coeficientes altos. Si esto cierto, es posible crear automáticamente un polinomio de evaluación en donde cada término represente un better-goal , con una definición inicial de coeficientes (tentativamente el mismo valor para cada término) y estudiar el comportamiento de nuestros better-goal por medio del proceso de ajuste de coeficientes descrito, de tal manera que logremos una jerarquización de los better-goal según los resultados obtenidos del ajuste.

- Aprendizaje por inducción generando árboles de decisión.

El aprendizaje inductivo de árboles de decisión fue elaborado por Ross Quinlan [Quinlan 83]. Quinlan trabajó inicialmente para resolver juegos de finales de ajedrez. Un exposición actual y detallada sobre esta técnica de aprendizaje inductivo y otra más poderosa (la desarrollada por Michalski) puede hallarse en [Zdrahal 1987]. Lo que aquí interesa es mencionar cómo se puede utilizar esta técnica en los lenguajes de consejos. Para este caso se escogió AL1. Podemos pensar en reglas en Al1 de la forma:

Si C1 & C2 & ... & Cn intenta Consejo(X,BGx,HGx,MCx,MCy)

Si al menos tuvieramos la relación:

Si C1 & C2 & ... & Cn intenta Consejo(X,BGx,_,_,_)

Tendríamos ya una guía de como hallar el consejo. Por ejemplo podríamos tener algo asi como :

Si TORRE_DIVIDE & REYES_OPUESTOS

intenta Consejo RESTRINGE_AL_REY_OPUESTO (n moves)

Así pues a la manera de Quinlan pueden generarse reglas de decisión entre condiciones y metas (better-goals) según un cierto número de movimientos. Lo que seguiría es completar el consejo. Es decir generar el holding-goal y los moves-constraint. Si en vez de usar consejos, usamos planes, ya solo necesitamos generar el holding-goal y el move-contraint para x. Un maestro puede ayudar en este objetivo, pero es posible en algunos casos también generar el move-constraint automáticamente. Es posible hallar a su vez el move-constraint por medio de árboles de decisión.

- Modelo de Pitrat para jugar ajedrez.

El primer punto interesante de este trabajo es que se utilizó precisamente para ajedrez. El programa puede aprender de un solo ejemplo. Lo que se aprenden son reglas de la forma: Si la posición presenta ciertas características intenta un procedimiento que considera tan solo un subconjunto de movimientos legales. El programa aprendió efectivamente reglas (heurísticas) utilizadas en el ajedrez. Aprendió, por ejemplo, la regla para clavar piezas. El programa partió de la siguientes reglas ya dadas (aprendizaje por implantación) :

1.- Si hay movimientos de captura, consideralos.

2.- Si una pieza se halla amenazada, considera los movimientos que involucren mover a dicha pieza.

El programa juega considerando solo los movimientos indicados por las reglas que posee y utilizando el método minimax. La función de evaluación se basa únicamente en el balance del material.

Este mecnismo puede utilizarse para obtener consejos inicialmente en forma de reglas. En este caso en vez de realizar una búsqueda minimax para ganar material se puede buscar la obtención de un better-goal.

Conclusiones.

Los lenguajes de consejos AL1 y AL3 contienen cada uno de ellos suficientes cualidades para utilizarse para describir y ejecutar estrategias en ajedrez. Conviene sin embargo utilizar AL1 en casos más simples (como finales básicos) y AL3 para los casos más complejos (finales difíciles o medio juego). La estructura general de ambos no tiene particularidades que les impidan utilizarse en tareas más generales que el Ajedrez. Sin embargo ámbos se apoyan en el concepto de "consejo" el cual si parece tener orientación a juegos con dos adversarios (como es el caso del ajedrez). La generalización de este concepto afectaría a estos dos lenguajes y les daría más fuerza. El lenguaje Prolog sin ser un requisito para desarrollar estos lenguajes si parece muy apropiado, por los elementos lógicos que utilizan AL1 y principalmente AL3.

Posibilidades de desarrollo muy generales de los lenguajes de consejos

Aprendizaje

Puede considerarse el aprendizaje como una parte indispensable de un sistema resolvedor de problemas, cuando menos desde un punto de vista humano. Sin embargo en Inteligencia Artificial es todavia difícil hallar máquinas o programas que aprendan en un nivel significativo. Las consideraciones del capítilo 4 pueden tomarse como un punto de partida.

Hacia una modelación del pensamiento.

Mediante AL3 se puede intentar construir una teoría que explique como juegan los humanos al ajedrez. Ya Newell Y Simon [Crosson 70] trabajaron en este sentido (en un lenguaje que construyeron y llamaron LPI-V) y crearon una teoría sobre cómo hallan los ajedrecistas (expertos) 'combinaciones' que imponen la pérdida de una pieza o un jaque mate al contrario. Esta teoría postula entre sus principales puntos que los jugadores usan la heurística del 'menor número de respuestas' como guía en su análisis. El punto principal es que tal heurística no es exclusiva del ajedrez sino que su uso es general en cualquier campo de la actividad humana. Esto da la pauta para proponerla como una posible 'regla humana' para la solución de problemas.

El tener pues, una serie de teorías específicas del procesamiento de información, permite especular en que poco a poco pueda inducirse una teoría general del procesamiento de la información y por tanto la posibilidad de modelarla y simularla en una computadora será factible aunque esto se haga de forma un tanto 'primitiva' y 'rudimentaria'.

Es por ésto que los lenguajes como AL3 tienen un valor muy apreciable en el campo del procesamiento de información.

A P E N D I C E

Mate de torres en Convert y una aportación más general

En uno de las primeras fases del trabajo de tesis, en la búsqueda de un lenguaje apropiado para definir estrategias en Ajedrez, contemplé a Convert (con alguna posible variante) como un posible punto de partida. Esto fue debido a que Convert es un lenguaje de reglas en base a patrones [Cisneros y McIntosh 1986] precisamente la manera que consideramos conveniente para modelar al experto humano. Como un paso inicial decidí programar el ejemplo de mate con torres y observar las conveniencias e incoveniencias. A continuación presento el código para este ejemplo para pasar posteriormente a comentar mi experiencia.

```
[rook.cnv]
[ Mauricio Osorio - Ricardo Valle Romo : June 15 th, 1986]
[ Chess ]
[Exclude DSK DIR BIO]
[[ROOK-KING vs. KING ENDGAME]]
[90 degree rotation]
((){}(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15)(
        (<(>(and,<[1]>,<0>)<)><<
                >><(>(and,<[1]>,<1>)<)><<
                >><(>(and,<[1]>,<2>)<)><<
                >><(>(and,<[1]>,<3>)<)><<
                >><(>(and,<[1]>,<4>)<)><<
                >><(>(and,<[1]>,<5>)<)><<
                >><(>(and,<[1]>,<6>)<)><<
                  (>(and,<[1]>,<7>)<)><<
                ,<(><0><1><2><3><4><5><6><7><)>);
        (<(>(and,<[1]>,<0>)<1><)><<
                >><(>(and,<[1]>,<2>)<3><)><<
                >><(>(and,<[1]>,<4>)<5><)><<
                >><(>(and,<[1]>,<6>)<7><)><<
                >><(>(and,<[1]>,<8>)<9><)><<
                >><(>(and,<[1]>,<10>)<11><)><<
                >><(>(and,<[1]>,<12>)<13><)><<
                >><(>(and,<[1]>,<14>)<15><)><<
                >>,(R,<(><1><)><<
                    >><(><3><)><<
                    >><(><5><)><<
                    >><(><7><)><<
```

```
                        >><(><9><)><<
                        >><(><11><)><<
                        >><(><13><)><<
                        >><(><15><)>)<<
                    >><(><0><2><4><6><8><10><12><14><)>);
)) R

[displays board]
(()()(0 1)(
        (<(><0><)><1>,(U,<1>)(%t,<0>));
)) U

[displays]
(()()(0 1)(
        (<0>:<1>,(%t,)(U,<0>)<0>:<1>);
        (<0>,(%t,)(U,<0>)<0>);
)) P

[read positions]
(()()(0 1 2)(
        (<0>;<1>;<2>;,<0><,><1><,><2>);
        (<0>;<1>;,(%t,give position for white rook)<0>;<1>;(%R);):
        (<0>;,(%t,give position for black king)<0>;(%R);):
        (<>,(%t,give position for white king)(%R);):
        (<0>,(%t,error)):
)) L

[set board]
(()()(0 1 2 3 4 5 6)(
        ((and,<:h:>,<0>)(and,<:d:>,<1>)<,><<
        >>(and,<:h:>,<2>)(and,<:d:>,<3>)<,><<
        >>(and,<:h:>,<4>)(and,<:d:>,<5>);<6><<
        >>,(P,(x,<5><,>(T,<4>)<,>T<,><<
        >>(x,<3><,>(T,<2>)<,>N<,><<
        >>(x,<1><,>(T,<0>)<,>B<,><6>))))<<
        >>);
        (<>,(L,);<<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
                >><(>........<)><<
        >>):
)) V

[Identifies general patterns]
(()()(0 1 2 9)(
        (<9>;(and,<:e:><:b:><:e:><:t:><-->,<0>)<<
                >>,<9>;(N,<0>));
        (<9>;(and,<:a:>(and,<(><:c:>,<1>)T<:c:>B<-->,<<
                >>(ITR,<1><[(#-,10-(&!,<1>))]>)<>,<0>)<<
                >>,<9>;(Y,(I,no reflex;<0>)));
```

```
        (<9>;(and,<:e:><(><:c:>T(and,<:c:>,<1>)<)><<
                >>(ITR,<(><[(#-,7-(&!,<1>))]>.<1><()>)<>,<0>)<<
                >>,<9>;(K,<0>));
        (R<9>;<0><>,<9>;(R,<0>)):
)) G

[first case, white king, rook, black king]
(()()(0 1 2 3 4 5)(
        ((and,<-->T(and,<:c:><)>,<1>)<-->N.<1><--><>,<0>)<<
                >>,(%t,move Rook)(0,<0>));
        ((and,<-->T.(and,<:c:><)>,<1>)<-->N<1><--><>,<0>)<<
                >>,(%t,move Rook)(0,<0>));
        (<1>(and,<:t:>,<2>)(and,<:l:><:e:>,<3>)<4><><<
                >>,(%t,bound enemy king)<1><3><2><4>);
        (<1>(and,<:b:>,<2>)(and,<:l:>,<3>)<4><><<
                >>,(%t,approach king)<1><3><2><4>);
        (<1>B.(and,<[20]><-->N<--><>,<2>)<<
                >>,(%t,approach king)<1>.B<2>);
        (<1>.B(and,<[18]><:c:><)><-><>,<2>)<<
                >>,(%t,approach king)<1>B.<2>);
        (<1>(and,<(><:c:>,<0>)B(and,<:c:><)>,<2>)(and,<[(&!,<0>)]>,<3>
                >>.(and,<[(&!,<2>)]>,<4>)<-->N<2><5><<
                >>,<1><0>B<2><(>.........<)><3>N<4><5><<
                >>(%t, opposition CHECK !!)<<
                >>(if,<5>,<>,:mate(%t,              MATE !!),));
        (<1><(>T.(and,<:c:><)><(>....<-->,<0>)<<
                >>,<1><(>.T<0>(%t,lose a tempo));
        (<1><(>.(and,<:c:>,<2>)T(and,<:c:><)><(>....<-->,<0>)<<
                >>,<1><(>T<2>.<0>(%t,lose a tempo));
        (<1>.T<)><0><<
                >>,<1>T.<)><0>(%t,lose a tempo));
        (<1>T(and,<:c:>,<2>)<)><0><<
                >>,<1><2>T<)><0>(%t,lose a tempo));
)) N

[moves rook]
(()()(0 1 2 3)(
        (<0><(>....(and,<:c:>,<1>)T(and,<:c:>,<2>)<)><3><<
                >>,<0><(>T....<1><2><)><3>);
        (<0>T(and,<:c:>,<1>)<)><2>,<0><1>T<)><2>);
)) O

[restores rotations]
(()()(0 1 2)(
        (F RR;<0>,<0>);
        (F<1>;<0>:<2>,<1>;(R,<0>):<2>):
        (R<1>;<0>,<1>;(R,<0>)):
        (;<0>,<0>);
)) H

[strategy for case 2]
(()()(0 1 2 3 8)(
        (<8>;<0>T(and,.<:c:>,<1>)B<2>,<8>;<0><1>TB<2>);
        (<8>;<:l:>(and,<:e:>,<1>)(and,<(><:c:>,<2>)TB<0>N<3><<
                >>,<8>;<1><2>B(C,(#-,8-(&!,<2>)):.........)<)><2>T.<0>
```

```
        (<8>;<0>TR(and,<:c:><)>,<1>)<:1:><2>N<3><<
                >>,<8>;<0>.B<1><0 T.<1><2>N<3>);
        (<8>;<0>TB<1><)><0>.<2>,<8>;<0>T.<1><)><0>B<2>);
        (no reflex;<0>,reflex;(W,<0>)):
)) I

[reflexes board]
(()()(0 1)(
        (<(><0><)><1>,(W,<1>)<(><0><)>);
)) W

[restores reflection]
(()()(0 1)(
        (reflex;<0>,(W,<0>));
        (no reflex;<0>,<0>):
)) X

[creates n blank squares]
(()()(0 1)(
        (<0>:(and,<[<0>]>,<1>)<--->,<1>);
)) C

[strategy case 3]
(()()(0 1 2 3)(
        (<0>T<1>B<2>N<3>.  ,(z,(&!,<0> )!<<
                >>(z,(&',<0>T<1>))!<<
                >>(z,(&!,<0>T<1>B<2>))<<
                >>!<0>T<1>B 2>N<3>));
        (<0 T<1>N<2>B<3>,(Z,(z,(&!,<0>))!<<
                >>(z,(&!,<0 T<1>N<2>))!<<
                >>(z,(&!,<0>T<1>))<<
                >'!<0>T<1>N<2>B<3>));
        )) K

[calculates coordinates]
(()()(0)(
        (<0>,(#+,(#/,<0>/10)+1)<,>(#+,(#%,<0>%10)+1));
)) z

[determines square for rook]
(()()(0 1 2 3 4 5 6)(
        (<0><,><1>!<2><,><3>!<4><,><5>!<6><<
                >>,(if,(y,<0><,><3>!<4><,><5>),True,<<
                >>(x,<0><,!<1><,>.<,>(x,<0><,><3><,>T<,><6>)),<<
                >>(x,<0><,><1><,>.<,>(x,<2><,><1><,>T<,><6>)))<<
                >>);
        )) Z

[verifies if the rook will be attacked in the desired square]
(()()(0 1 2 3)(
        (<0><,><1>!<2><,><3><<
                >>,(nf,(w,(#-,<2>-<0>))<,>(w,(#-,<3>-<1>)),<<
                >>(or,1,0)<,>(or,1,0),True,));
)) Y
```

```
[change position]
(()()(0 1 2 3 4)(
        (<0><,><1><,><2><,><<
        >>(and,<[(#-,(#+,(#*,(#-,<0>-1)*10)+<1>)-1)]>,<3>)<[11><4><<
t
        >>,<3><2><4>);
)) x

[absolute value]
(()()(0)(
        (-<0>,<0>);
        (<0>,<0>);
)) w

[moves]
(()()(0 1 2 3 4 5)(
        ((and,<:h:>,<0>)(and,<:d:>,<1>)-(and,<:h:>,<2>)(and,<:d:>,<3>)
                >>;<5>;<4><<
                >>,(x,<1><,>(T,<0>)<,>.<,>(x,<3><,>(T,<2>)<,><5><,><4>
        (quit;N;<4>,<4>:quit);
        (;<4>,(%t,your move, please: )(%R);N;<4>):
        (<0>:N;<4>,(%t,what is <0> ??);<4>):
)) M

[changes letter-coordinate to digit-coordinate]
(()()(0)(
        (<0>,(#+,(#-,(&d,(&u,<0>))-64)+1));
)) T

[game]
(()()(0 1)(
        (<-->:<0>,);
        (me;<0>,you;(P,(H,(G,RRRR;<0>)))):
        (you;<0>,me:(P,(M,;<0>))):
)) J

[main]
((
        ((ITR,<()(ITR,  -,.,N))<)>)) a
        (<()<:c:>2<:c:   >) b
        (<()<:c:>T<:c: * >) t
        ((ITR,.)) c
        (<(>........<)  1
        ((ITR,<:1:>)) e
        ((or,(IVL,A,H,),(IVL,a,h,))) h
        ((IVL,1,8,)) d
 )
(
  ((%t,next instruction )(%R,)) r
)(0 1)(
(read,(V,)):
(identify<0>,write(H,(G,RRRR;<0>))):
(write<0>,(P,<0>)):
(game<0>,(J,me;<0>))<
(move<0>,(nf,(M,;<0>),<0>:quit,write<=>,<0>)):
```

```
(end,);
(<(><O>,(r)<(><O>):
(<--><(><O>,(%t,error)<(><O>):
(,(r)):
))
[end]
```

Pocas horas de trabajo nos llevó (a Ricardo Valle , coautor del programa , y a mi) desarrollar este trabajo (diseño y programación) . La parte central del algoritmo ocupa 1 página de código. La definición de los patrones fue muy simple, a pesar de que utilizamos patrones unidimencionales para simular patrones bidimensionales. Vale aquí la pena señalar que sería muy ventajoso definir algún lenguaje que permitiera manejar directamente patrones bidimensionales. Lo que resulto más incomodo fue definir las acciones tomadas. Fueron muy 'procedurales' y parecía requerirse una capacidad 'declarativa' de expresión . El 'consejo' o 'plan' se ajustan muy bien a esta demanda. Por otro lado en Ali no queda claro como definir los patrones y como una vía a seguir se le deja esta tarea a un segundo lenguaje sobre el cual se halla superpuesto Ali. PROLOG resultó un lenguaje apropiado para este cometido. Queda sin embargo abierta la posibilidad de utilizar Convert para este proposito (en lo particular, tanto en algunas casos me fue más comodo definir estos patrones en Convert, como en otros en PROLOG). Quizá lo mejor sea definir un lenguaje apropiado para definir patrones en Ajedrez. Pero seguramente para este propósito va a ser muy significativo tomar en cuenta las diferentes formas de definir patrones y en particular yo tendría en cuenta a Convert.

Programación de uso.

Para usar el sistema AL3 se requiere:

a) La base de conocimientos.

b) Un módulo de los predicados usados en la base de conocimientos.

c) Un módulo de la entrada y salida según el problema.

Una vez que se tienen estos módulos se ejecuta el programa con el comando :

```
PROLOG LOAD AL3_MAIN
```

La pregunta que debe efectuarse a PROLOG es :

```
?((INICIA))
```

Entonces el programa preguntará por el problema a resolver.
A continuación se presenta un ejemplo de base de conocimientos y algunas corridas. La base es la misma que la expuesta en el capítulo tres , pero en esta ocasión con la sintaxis que interpreta esta versión de AL3.

```
((COMENTARIO BASE DE CONOCIMIENTOS para final de rey y peón vs. rey
        CONCEPTOS DE ALTO NIVEL
))

((COMENTARIO DEFS))
((LEMAS (PIERDE_PEON CORRE_PEON CASILLAS_CRITICAS CASILLAS_CRITICAS_T
         PEON_BLOQUEADO  )))

((OBSERVACIONES (PEON_T) ))

((METODO M0 FACIL
        ((H1 : GANA)
         (PEON_T = 0))

        (     ((H2 : GANA_PEON_NT))
              (H2 > H1)
        )        ))

((METODO M1 FACIL
        ((H1 : GANA)
         (PEON_T = 1))

        (      ((H2 : GANA_PEON_T))
               (H2 > H1)
        )        ))

((METODO M2 FACIL
        ((H1 : GANA_PEON_NT))

        (     ((H2 : CORRE_PEON)
              (H3 : CASILLAS_CRITICAS)
              (H4 : PATRON_BP6))

                ((H2 + (H3 + H4)) > H1)
        )         ))

((METODO M3 FACIL
        ((H1 : GANA))

        (        ((H  : TABLAS))
                 (H1 = (~ H2) )
        )        ))

((METODO M4 FACIL
        ((H1 : TABLAS))
            (    ((H2 : PIERDE_PEON)
                  (H  : PEON_BLOQUEADO))
                 ((H2 + H3) >  H1)
            ) ))
```

```
((METODO M5 DIFICIL
        ((H1 : GANA_PEON_NT))
        (    ((H2 : (SUC P_BLANCO_NT) ))
               (H2 > H1)
        ) ))
((METODO M6 DIFICIL
        ((H1 : GANA_PEON_T))
        (    ((H2 : (SUC P_BLANCO_T)))
               (H2 > H1)
        ) ))

((METODO M7 FACIL
        ((H1 : (SUC P_BLANCO_NT)))

        (       ((H2 : (REFUTA P_NEGRO P_BLANCO_NT)))

                       (H2 > (~ H1))
            ) ))

((METODO M8 FACIL
        ((H1 : (SUC P_CORONA_T)))

        (       ((H2 : (REFUTA P_NEGRO P_BLANCO_NT)))

                       (H2 > (~ H1))
            ) ))

((METODO M9 DIFICIL
        ((H1 : TABLAS))
        (    ((H2 : (SUC P_NEGRO) ))
               (H2 > H1)
        ) ))

((COMENTARIO    CADA plan ES UN QUINTUPLE
                       PLAN PARA                X
                       LIMITE DE EXPANCION
                       OBJETIVO A ALCANZAR      "BETTER GOAL"
                       OBJETIVOS A MANTENER     "HOLDING GOAL"
                       SUBCONJUNTO DE MOVS X    "MOVE CONSTRAIN"
))
 ((PLAN P_BLANCO_T  (
                BLANCAS
                1
                ((CASILLAS_CRITICAS_T P2) + (CORRE_PEON P2))
                ((PEON_A_SALVO P2) & (NOAHOGADO P2))
                ((MREYB P1 P2) & (RB_CERCA P1 P2))
            )))
 ((PLAN P_BLANCO_NT (
                BLANCAS
                1
                (((CASILLAS_CRITICAS P2) + (CORRE_PEON P2)) + (PATRON
                ((PEON_A_SALVO P2) & (NOAHOGADO P2))
                ((MREYB P1 P2) & (RB_CERCA P1 P2))
           )))
```

```
((PLAN P_NEGRO (
                NEGRAS
                1
                ((PIERDE_PEON P2) + (PEON_BLOQUEADO P2))
                ()
                (RN_CERCA P1 P2)
        )))
```

Ejemplo

Se interroga a PROLOG :

```
?((INICIA))

(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
UBICACION DE REY NEGRO
(5 5)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . N . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
UBICACION DE REY BLANCO
(5 2)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. B . . N . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. . . . . .)
UBICACION DE PEON BLANCO
(3 4)
(. . . . . .)
(. . . . . .)
(. . . . . .)
(. B . . N . . .)
(. . . . . .)
(. . . P . . . .)
(. . . . . .)
(. . . . . .)
PREGUNTA: GANA
"INICIO DE RESOLUCION "

"OBSERVA " ((PEON_T 0))

"HIPOTESIS INTERESANTE " (3 GANA)
"  ELECCION " (METODO M0)

"HIPOTESIS INTERESANTE " (4 GANA_PEON_NT)
"  ELECCION " (METODO M2)
```

```
"HIPOTESIS INTERESANTE " (5 CORRE_PEON)
"   ELECCION " (LEMA CORRE_PEON)

"HIPOTESIS INTERESANTE " (6 CASILLAS_CRITICAS)
"   ELECCION " (LEMA CASILLAS_CRITICAS)

"HIPOTESIS INTERESANTE " (7 PATRON_BP6)
"HIPOTESIS INTERESANTE " (3 GANA)
"   ELECCION " (METODO M3)

"HIPOTESIS INTERESANTE " (8 TABLAS)
"   ELECCION " (METODO M4)

"HIPOTESIS INTERESANTE " (9 PIERDE_PEON)
"   ELECCION " (LEMA PIERDE_PEON)

"HIPOTESIS INTERESANTE " (10 PEON_BLOQUEADO)
"   ELECCION " (LEMA PEON_BLOQUEADO)

"HIPOTESIS INTERESANTE " (7 PATRON_BP6)
"HIPOTESIS INTERESANTE " (3 GANA)
"HIPOTESIS INTERESANTE " (4 GANA_PEON_NT)
"   ELECCION " (METODO M5)

"HIPOTESIS INTERESANTE " (11 (SUC P_BLANCO_NT))
"   ELECCION " (METODO M7)

"HIPOTESIS INTERESANTE " (12 (REFUTA P_NEGRO P_BLANCO_NT))
"   ELECCION " (REFUTA P_NEGRO P_BLANCO_NT)

"REFUTA RESULTO " 0
"HIPOTESIS INTERESANTE " (7 PATRON_BP6)
"HIPOTESIS INTERESANTE " (3 GANA)
"HIPOTESIS INTERESANTE " (4 GANA_PEON_NT)
"HIPOTESIS INTERESANTE " (8 TABLAS)
"   ELECCION " (METODO M9)

"HIPOTESIS INTERESANTE " (13 (SUC P_NEGRO))
"   ELECCION " (PLAN P_NEGRO)

"PLAN RESULTO " 0
"HIPOTESIS INTERESANTE " (11 (SUC P_BLANCO_NT))
"   ELECCION " (PLAN P_BLANCO_NT)

"PLAN RESULTO " 1
```

L I S T A D O S

```
RESPUESTA  (SI)

JUSTIFICACION

TABLAS   RESULTO  FALSO

GANA_PEON_NT   RESULTO  CIERTO

GANA   RESULTO  CIERTO

(SUC P_BLANCO_NT)   RESULTO  CIERTO

(SUC P_NEGRO)   RESULTO  FALSO

(REFUTA P_NEGRO P_BLANCO_NT)   RESULTO  FALSO

PEON_BLOQUEADO   RESULTO  FALSO

PIERDE_PEON   RESULTO  FALSO

CASILLAS_CRITICAS   RESULTO  FALSO

CORRE_PEON   RESULTO  FALSO

PEON_T   RESULTO  FALSO
```

```
((COMENTARIO AL3_MAIN.LOG , procedimiento principal))

((COMENTARIO ***  PROCEDIMIENTO GENERAL ))

((INICIA)
        (TRA _BIBLIOT)
        (SOL_BASE)
)

((COMENTARIO RESUELVE SEGUN BASE DE ENTRADA))

((SOL_BASE)
        (LEE_BASE)
        (KILL COMENTARIO)
        (TRADUCE_METODOS)
        (PROBLEMA)
)

((COMENTARIO    Entrada  NO HAY <el sistema interroga
                                interroga por posición inicial
                                        y por query
                Sal da   NO HAY <el sistema responde>
                                responde SI o NO o
                                        INSUFICIENTES RECURSOS
                                proporciona justificación
))
((PROBLEMA)
        (INICIAL_POS Z9)
        (P "PREGUNTA: ")  (R X7)
        (INICIAL_BASE_CON X7 Z)
        (INICIAL_RECURS X)
        (PP "INICIO DE RESOLUCION ")(PP)
        (SOLUCIONA Z9 Z X x X9)
        (PP)
        (P "RESPUESTA " x)(PP)(PP)
        (PP JUSTIFICACION)  (REPORTA_JUSTIF X9)
)

((SOLUCIONA Z9 X Z x X9)
        (OBSERVACIONES Y)
        (EJECUTA_OBS Y Z9 X X1)
        (RESUELVE Z9 X1 Z x X9)
)

((COMENTARIO Entrada     Z9 Posición
                         X  Base Conocimientos Corriente
                         Z  Recursos
))
((RESUELVE Z9 X Z (SI) X9)
        (TARGET X X1)
        (PRUEBA X1)
        (HECHOS_CORR X X9)
)
```

```
((RESUELVE Z9 X Z  (NO) X9)
        (TARGET_NEG X X2)
        (PRUEBA X2)
        (HECHOS_CORR X X9)
)
((RESUELVE Z9 X Y x X9)
        (IF (CONSULTA_BASE_CONOC Z9 X Y X1 Y1)
           ((RESUELVE Z9 X1 Y1 x X9) / )
           ((EQ x (INSUFICIENTES RECURSOS PARA RESPONDER))
                  (HECHOS_CORR X X9) )
        )
)
((COMENTARIO Entrada:    Z9 es posición
                         X es base corriente de conocimientos
                         Y son parte de los recursos, o sea
                              métodos
             Salida :    X1,Y1 son X,Y actualizadas
))
((CONSULTA_BASE_CONOC Z9 X Y X2 Y1)
        (SELECCION_PROC X Y  Z)
        (PP "  ELECCION "  Z)(PP)
        (EJECUTA  Z Z9 X Y X1 Y1)/
        (ACTUALIZA_BASE_CORR X1 X2)
)
((COMENTARIO CARGA BIBLIOTECA))
((TRAE_BIBLIOT)
        (LOAD ERRTRAP)
        (LOAD AL3_CONS)
        (LOAD AL3_TRAD)
        (LOAD AL3_CKP)
        (LOAD AL3_GETP)
        (LOAD AL3_PWN2)
        (LOAD AL3_RESO)
        (LOAD AL3_TARG)
        (LOAD AL3_EXEC)
        (LOAD ID_PWN)
)

((COMENTARIO LEE BASE DE CONOCIMIENTOS))

((LEE_BASE)
        (PP ARCHIVO BASE DE CONOCIMIENTOS .LOG)
        (R X)
        (LOAD X)
)
```

```
((COMENTARIO  Modulo AL3_GETP.LOG que selecciona el procedimiento
     a utilizar en un ciclo de ejecución , por procedimiento
          se tiene
                Lemas
                Métodos <simples o difíciles>
                Refutaciones
                Planes
))

((COMENTARIO  Entradas  X. es la base corriente de conocimientos
                        Y  son los recursos
              Salida    Z  es el proc escogido <caso,nombre>
))
((SELECCION_PROC X Y Z)
        (OBTEN_H_INTERES X z9)
        (PP "HIPOTESIS INTERESANTE " z9)
        (ESCOGE_PROC z9 Y X Z)
)

((COMENTARIO Entrada    X es la base corriente de conocimientos
             Salida     y es la hipótesis mas interesante
))
((OBTEN_H_INTERES X y)
        (HIPOTESIS_CORR X Y1)
        (TARGET X Z)
        (MEJOR_HIP Y1 Z y)
)
((COMENTARIO Entrada    Y1 Hipótesis corrientes
                        Z  Target
             Salida     y  Hipótesis escogida
  NOTA : versión preliminar
))
((MEJOR_HIP Y1 Z y)
        (ON y Y1)
)

((COMENTARIO *** ESCOGE PROCEDIMIENTO *** ))
((COMENTARIO Entradas   z9 hipótesis interesante
                        Y recursos
                        Z7 base corr
             Salidas    Z  procedimiento <nombre>
))
((ESCOGE_PROC z9 Y Z7 (LEMA Z) )
        (ES_LEMA z9 Y Z)
)
((ESCOGE_PROC z9 Y (Z5 Z6|Z9) (METODO Z) )
        (CHECA_METODO_SIMPLE z9 Y Z (Z5 Z6))
)
((COMENTARIO META_PLAN))
((ESCOGE_PROC z9 Y (Z5 Z6|Z9)  Z)
        (CHECA_META_PLAN Z  z9 (Z5 Z6))
)
```

```
((ESCOGE_PROC (x (REFUTA ! Z9)) Y Z7 (REFUTA ! Z9) ))
((ESCOGE_PROC z9 Y Z7 (PLAN Z))
        (ES_PLAN z9 Y Z)
)

((COMENTARIO PLAN DIFICIL))
((ESCOGE_PROC z9 Y (Z5 Z6¡Z9) (METODO Z))
        (CHECA_METODO_DIFICIL z9 Y Z (Z5 Z6))
)

((COMENTARIO MODULO AL?_EXEC.LOG de ejecución de procedimientos
                LEMAS,METODOS Y PLANES
))

((COMENTARIO Entrada : X LISTA DE OBSERVACIONES
                       Z9 POSICION
                       Z  BASE
             Salida  : Z1 BASE ACTUALIZADA
))
((EJECUTA_OBS X Z9 Z Z1)
        (ISALL Y (x z)
                (ON x X)
                (VALOR_LEMA x Z9 z)
        )
        (PP "OBSERVA " Y)(PP)
        (ALTA_BASE Z (Y () () ) Z1)
)

((COMENTARIO    Entrada Z nombre del LEMA
                        Z9 posición
                        X  base conocimientos corriente
                        Y  recursos
                Salida  X1, X actualizado
))
((EJECUTA (LEMA Z) Z9 X Y X1 Y)
        (VALOR_LEMA Z Z9 z)
        (ALTA_BASE X
                    ( ((Z z)) () () )
                   X1 )
)
((EJECUTA (METODO Z) Z9 X Y X1 Y1)
        (TOMA_ACCION_METOD Z Z1)
        (ALTA_BASE X Z1 X1)
        (ELIMINA_METODO Z Y Y1)
)
((EJECUTA (PLAN Z) Z9 X Y X1 Y)
        (VALOR_PLAN Z Z9 z)
        (PP "PLAN RESULTO " z)
        (ALTA_BASE X
                   ( (((SUC Z) z)) () () )
                   X1)
)
```

```
((EJECUTA (REFUTA ! Z) Z9 X Y X1 Y)
        (VALOR_REFUTA Z Z9 z)
        (PP "REFUTA RESULTO " z)
        (ALTA_BASE X
                   ( (( (REFUTA ! Z) z)) () () )
                   X1)
)

((COMENTARIO METAPLANES))
((EJECUTA (META_PLAN_OR (Z Z1 Z2)) Z9 X Y X1 Y)
        (EQ Y9 (REFUTA  (Z1 OR Z2) Z) )
        (ALTA_BASE X
                   (
                     ()
                     (Y9)
                     (
                       ( (1) (Y9) )
                       ( (1) ((SUC Z)) )
                     )

                     X1)
)

((EJECUTA (META_PLAN_MOD1 (Z Z1 Z2)) Z9 X Y X1 Y)
        (EQ Y9 (SUC (Z MOD1 Z2)) )
        (ALTA_BASE X
                   (
                     ()
                     (Y9)
                     (
                       ( (1) (Y9) )
                       ( (Z1) (0) )
                     )
                    )
                     X1)
)


((VALOR_PLAN  Z Z9 z)
        (ACCESA_PLAN Z Z5)
        (APPEND Z5 (()) Z6)
        (IF (FORCING_TREE  Z9 Z6 Z8)
                ((EQ z 1))
                ((EQ z 0))
        )
)
((VALOR_LEMA Z Z9 z)
        (IF (Z Z9)
                ((EQ z 1))
                ((EQ z 0))
        )
)
```

```
((COMENTARIO Z1 REFUTA A Z2 ?))
((VALOR_REFUTA (Z1 Z2) Z9 z)
        (ACCESA_PLAN Z1 (X y1 Y1 Y2 Y3))
        (ACCESA_PLAN Z2 (X6 x1 X1 X2 X3))
        (MAX x1 y1 z1)
        (IF (FORCING_TREE Z9 (X6 z1 (X1 + (~ Y2))  X2 X3 Y3) Z8)
                ((EQ z 0))
                ((EQ z 1))
        )
```

```
((COMENTARIO JUEGA SEGUN CONSEJO
        X1 ES POSICION DE ENTRADA Y X2 DE SALIDA
        Z ES CONSEJO))
((COMENTARIO GRAFICA AND-OR O TERMINOS DE CONSEJOS VERIFICA SI SE
         SATISFACE CONSEJO))
((COMENTARIO X7 ES POSICION INICIAL, X1 POSICION CORRIENTE, X2 POS SIG .
        JUEGA Y7))

((FORCING_TREE X7 Z  X2)
        (F_TREE X7 X7 Z X2)
)
((COMENTARIO JUEGA Y7 y EL CONSEJO ES PARA Y7))
((F_TREE X7 (Y7 ! X1) (Y7 x X Y Z Z1)  X2)
        (MOVLEGAL (Y7 !X1) X2)
        (SATISF_FUN_LOG Z (X7 (Y7!X1) X2))
        (SATISF_FUN_LOG Y (X7 (Y7!X1) X2))
        (IF (SATISF_FUN_LOG X (X7 (Y7!X1) X2))
                ((CIERTO))
                ((LESS 1 x)
                 (F_TREE X7 X2 (Y7 x X Y Z Z1) X3)
                )))

((EL CONSEJO ES PARA Y6 PERO NO JUEGA EL PRIMERO
        SE DEBE AGREGAR QUE SE CUMPLA EL HOALDING GOAL ?))
((F_TREE X7 X1 (Y6 x X Y Z Z1)  X2)
        (NOT EQ X1 (Y6!Y9))
        (SUM x1 1 x)
        (FORALL ((MOVLEGAL X1 X2)
                  (SATISF_FUN_LOG Z1 (X7 X1 X2))  )
                ((F_TREE X7 X2 (Y6 x1 X Y Z Z1) X3))
        ))

((SATISF_FUN_LOG (X1 + X2) x)
        (OR ((SATISF_FUN_LOG X1 x))
            ((SATISF_FUN_LOG X2 x))
        )
)
((SATISF_FUN_LOG (X1 & X2) x)
        (SATISF_FUN_LOG X1 x)
        (SATISF_FUN_LOG X2 x)
)
((SATISF_FUN_LOG (~  X1) x)
        (NOT SATISF_FUN_LOG X1 x)
)
((SATISF_FUN_LOG (X P1 P2) (Z Z1 Z2))
        (X Z1 Z2)
)
((SATISF_FUN_LOG (X P1 P2) (Z Z1 Z2))
        (X Z Z2)
)
((SATISF_FUN_LOG (X · 2) (Z Z1 Z2))
        (X Z2)
)
```

```
((SATISF_FUN_LOG (X PI) (Z Z1 Z2))
        (X Z)
)
((SATISF_FUN_LOG () Z ))
```

```
((COMENTARIO MODULO AL3_CMP.LOG
        ACCESO A BASE CORRIENTE DE CONOCIMIENTO
        LA BASE ES UN CUADRUPLE <HE HI T N> Donde
                HE es una lista de hechos
                HI es una lista de hipótesis
                T es el target <incluyendo al negado>
                N número CLAVE corriente de hipótesis
))
((COMENTARIO    Entrada Y hipótesis inicial
                Salida  BASE INICIAL
))
((INICIAL_BASE_CON Y (() ((3 Y)) (Z1 Z2) 3))
        (EQ Z1 (((1) (3))) )
        (EQ Z2 (((3) (0))) )
)
((COMENTARIO ACTUALIZA BASE))
((ACTUALIZA_BASE_CORR (X1 X2 Y1 y) (Z1 Z2 Y2 y) )
        (ACTUALIZA_TARGET Y1 Y2 Z)
        (CLAVES_A_NOMBRES Z X2 Z5)
        (APPEND Z5 X1 Z1)
        (ELIMINA_HIPOTESIS X2 Z5 Z2)
)
((COMENTARIO Entrada    X es la base de con
             Salida     Y son los hechos de la base
))
((HECHOS_CORR (X!X9) X))
((COMENTARIO Entrada    X es la base de con
             Salida     Y son las hipótesis de la base
))
((HIPOTESIS_CORR (X1 X!X9) X))
((COMENTARIO Acceso a los targets))
((TARGET (X1 X2 (Y1 Y2) Z) Y1))
((TARGET_NEG (X1 X2 (Y1 Y2) Z) Y2))

((COMENTARIO ALTA A BASE CORRIENTE
        Entrada  X1 HECHOS NUEVOS
                 X2 HIPOTESIS
                 X3 FUNCION LOGICA <ARGS,FUNC>
                 BASE CORRINTE
        Salida   Base nueva
))
((ALTA_BASE (Y1 Y2 Y3 y) (X1 () ()) (Y5 Y6 Y7 y))
        (APPEND X1 Y1 Y5)
        (ELIMINA_HIPOTESIS Y2 X1 Y6)
        (REDEF_HECHOS  X1 Y2 X7)
        (SUBS_VALS_TARGET X7 Y3 Y7)
)
((ALTA_BASE (Y1 Y2 Y3 y) (() X2 X3) (Y1 Y6 Y7 y1))
        (DEFINE_CLAVES_HIP X2 y Y2 X5 y1)
        (UNION X5 Y2 Y6)
        (REDEF_FUN Y6 X3 Y7)
        (PROD_TARGS Y2 Y3 Y7)
)
```

```
((COMENTARIO    Entrada Y1 Lista de hipótesis
                        X  Lista de hechos
                Salida  Y2 Nueva lista de hipótesis, es decir
                           se eliminan los hechos ..
        NOTA: FUNCIONA CASO NULO ? EL ULTIMO ISALL ES PARA REVERSE
))
((ELIMINA_HIPOTESIS Y1 X Y2)
        (ISALL Y5 (x1 x)
                (ON (x1 x) Y1)
                (NOT ON (x x2) X)
        )
        (ISALL Y2 y
                (ON y Y5)
        )
)
((COMENTARIO Entrada X  Lista de hechos <nombres>
                     Y  Lista de hipótesis
             Salida  Y1 Lista de hechos <claves>
))
((REDEF_HECHOS X Y X1)
        (ISALL X1 (x x1)
                (ON (y x1) X)
                (ON (x y) Y)
        )
)
((COMENTARIO    Entrada  <x4 X2> Lista de hipótesis nuevas
                         y       Clave corriente
                         Y2      Hipótesis anteriores
                Salida   <x5 X5> Nuevas hipótesis
                         y1      Clave nueva
))
((DEFINE_CLAVES_HIP () y Y2 () y))
((DEFINE_CLAVES_HIP (x4|X2) y Y2 (x5|X5) y1)
        (DEFINE_CLAVE x4 y Y2 x5 y6)/
        (DEFINE_CLAVES_HIP X2 y6 Y2 X5 y1)
)
((COMENTARIO Entrada    y1 Hipótesis nueva
                        z5      Clave corriente
                        Z       Hipótesis anteriores
             Salida     <x2 y1>  Hipótesis con clave
))
((DEFINE_CLAVE  y1 z5 Z (x2 y1) z5)
        (ON (x2 y1) Z)
)
((DEFINE_CLAVE  y1 z5 Z (z6 y1) z6)
        (SUM z5 1 z6)
)
((COMENTARIO CAMBIA FUNCION DE HIPOTESIS NOMBRES A HIPOTESIS CLAVES
        NOTA : FUNCION INEFICIENTE
))
((REDEF_FUN Z  X1 X2)
        (CONMUTA_PARES Z Z1)
        (SUBS_VALS_T Z1 X1 X2)
)
```

```
((COMENTARIO CONVIERTE HECHOS DE CLAVES A NOMBRES))
((CLAVES_A_NOMBRES X Y X1)
        (ISALL X1 (x y1)
                (ON (x1 y1) X)
                (ON (x1 x) Y)
        )
)

         ((COMENTARIO MODULO AL3_RESO.LOG  DE ACCESO A RECURSOS
        Los recursos son un cuadruple
                <x1 y1 y2 z> donde
                        x1 es lista de lemas
                        y1 es lista de métodos simples
                        y2 es lista de métodos difíciles
                        z  es lista de planes
))

((COMENTARIO Entrada    DE LA BASE DE CONOC
             Salida     RECURSOS (X Y1 Y2 Z) donde
                                X  son lemas
                                Y1 métodos simples
                                Y2 métodos difíciles
                                Z planes
))
((INICIAL_RECURS (Y1 Y2) )
        (METODOS_SIMPLES Y1)
        (METODOS_DIFICILES Y2)
)
((COMENTARIO Entrada    z1 hipótesis clave <según target>
                        z2 hipótesis nombre
                        Y1 LEMAS
                        Y9 RECURSOS <METODOS>
             Salida     cuando es LEMA
))
((ES_LEMA (z1  2) Y9 z2)
        (LEMAS Y1)
        (ON z1  Y1)
)
((COMENTARIO Entrada    z1 hipótesis clave <según target>
                        z2 hipótesis nombre
                        X  PLANES
                        Y RECURSOS <METODOS>
             Salida     cuando es PLAN
))
((ES_PLAN (z1 (SUC z2)) Y z2))

((COMENTARIO ACCESA ELEMENTOS DE PLAN
        VERSION PRELIMINAR))
((ACCESA_PLAN (X1 OR X2) (Z y (Z1 + Z6) (Z2 + Z7) (Z3 + Z8) ) )
        (ACCESA_PLAN X1 (Z z Z1 Z2 Z3))
        (ACCESA_PLAN X2 (Z5 z2 Z6 Z7 Z8))
        (MAX z z2 y)
)
```

```
((ACCESA_PLAN X (X1 y Y1 Y2 Y3))
        (NOT EQ X (X7 OR X8))
        (PLAN X (X1 y Y1 Y2 Y3|Y9))
)
((COMENTARIO ** ACCESO A METODOS DE LOS RECURSOS **
                Un método es un TRIPLE (N P A ) donde
                        N es NOMBRE del método
                        P es PRECONDICION
                        A es ACCION <HE HI F>
))
((COMENTARIO Entrada    z1 hipótesis clave <según target>
                        z2 hipótesis nombre
                        Y1 METODOS SIMPLES
                        Y9 RECURSOS RESTANTES
             Salida     Z  Nombre del METODO SIMPLE

))
((CHECA_METODO_SIMPLE (z1 z2) (Y1 Y9) Z Z7)
        (ON Z Y1)
        (CHECA_METODO z2 Z Z7)
)
((COMENTARIO Entrada    z1 hipótesis clave <según target>
                        z2 hipótesis nombre
                        Y1 METODOS DIFICILES
                        Y9 RECURSOS RESTANTES
                        Z7 hechos e hipótesis de la base corr
             Salida     Z  Nombre del METODO DIFICIL

))
((CHECA_METODO_DIFICIL (z1 z2) (Y9 Y1) Z Z7)
        (ON Z Y1)
        (CHECA_METODO z2 Z Z7)
)
((COMENTARIO Entrada    z hipótesis nombre
                        Y Método
                        (Z1 Z2) hechos e hipótesis de la base corr
             Salida     si se satisface
        NOTA: QUE SE HACE CON LAS PRECONDICIONES HIPOTESIS
))
((CHECA_METODO z Y (Z1 Z2))
        (TOMA_PRECOND_METOD Y (Y1 Y2) )
        (ON z Y2)
        (SUBSET Y1 Z1)
)

((COMENTARIO DA DE BAJA METODO))
((ELIMINA_METODO x (X1 X2) (X3 X4))
        (QUITA_x x X1 X3)
        (QUITA_x x X2 X4)
)
((COMENTARIO ACCESO A PRECONDICIONES
                Y es NOMBRE del método
                z es PRECONDICION
))
```

```
((TOMA_PRECOND_METOD Y z)
        (METODO Y z1 Z)
)
((TOMA_ACCION_METOD Y z)
        (METODO Y z1 z)
)


((COMENTARIO Entrada    z1 hipótesis clave <según target>
                        z2 hipótesis nombre
                        Z1 Z2 hechos e hipótesis de la base corr
             Salida     <CASO Z>  Nombre del META_PLAN
                        Z ES CONTEXTO

))
((CHECA_META_PLAN (META_PLAN_OR Y X1 X2)) (z1 (SUC Y))  (Z1 Z2))
        (ON ((REFUTA X1 Y) 0) Z1
        (ON ((REFUTA X2 Y) 0) Z1
        (VALID_OR_COMB X1 X2 Y z1 Z2))
)

((COMENTARIO VERIFICA META-OR CON SENTIDO ))
((VALID_OR_COMB X Y Z9 (Z1 Z2))
        (PLANES_PRIMITIVOS X X1)/
        (PLANES_PRIMITIVOS Y Y1)/
        (NOT SUBSET X1 Y1)
        (NOT SUBSET Y1 X1)
        (NOT ON ((REFUTA (X OR Y) Z9) z) Z1)
        (NOT ON ((REFUTA (Y OR X) Z9) z1) Z1)
        (NOT ON (x7 (REFUTA (X9 OR Y9) Z9)) Z2)
)
((PLANES_PRIMITIVOS (X OR Y) Z)
        (PLANES_PRIMITIVOS X X1)
        (PLANES_PRIMITIVOS Y Y1)
        (APPEND X1 Y1 Z)
)
((PLANES_PRIMITIVOS X (X) ))
```

```
((COMENTARIO manejo de targets))

((COMENTARIO Entrada : X1 es el target, X2 es el target negado
             Salida  : Y es la pareja de targets actualizada
                       Z6 son hechos encontrados, es decir
                        hipótesis con sus valores determinados
))

((ACTUALIZA_TARGET  (X1 X2) Y Z6)
        (SIMPLIFICA_TARG X1 Z1)
        (SIMPLIFICA_TARG X2 Z2)
        (DEDUCE_VALORES (Z1 Z2) Y Z6)
)

((SIMPLIFICA_TARG X1 Y1)
        (BORRA_TAUTOL X1 X2)
        (BORRA_LINEAS_IMPLICADAS X2 Y1)
)

((COMENTARIO PRODUCTO DE TARGETS
        Entrada X es TARGET que se agrega
                (Y1 Y2) TARGETS de la base
        Salida  (Y2 Z2) TARGETS de la base actualizada
))
((PROD_TARGS X (Y1 Z1) (Y2 Z2))
        (PROD X Y1 Y2)
        (PROD X Z1 Z2)
)
((COMENTARIO Z es el producto del target X por el taget Y))
((PROD X Y Z)
        (ISALL Z (z1 z2)
                (ON (x1 x2) X)
                (ON (y1 y2) Y)
                (UNION x1 y1 z1)
                (UNION x2 y2 z2)
        )
)

((COMENTARIO Un target es una prueba))
((PRUEBA ()))
((PRUEBA (x|X))
        (TAUTO  x)
        (PRUEB . X)
)

((COMENTARIO Una línea es una tautología))
((TAUTOL (X Y))
        (ON 0 X)
)
((TAUTOL (X Y))
        (ON 1 Y)
)
```

```
((TAUTOL (X Y))
        (ON x ))
        (ON x Y)
)

((COMENTARIO Elimina lineas_tautologías de un target))
((BORRA_TAUTOL () () ))
((BORRA_TAUTOL (x!X) Y)
        (IF (TAUTOL x)
            ((BORRA_TAUTOL X Y))
            ((BORRA_TAUTOL X Y1)
             (EQ Y (x!Y1)) )
        )
)

((COMENTARIO Elimina de un target líneas implicadas por otras))
((BORRA_LINEAS_IMPLICADAS () () ))
 BORRA_LINEAS_IMPLICADAS (x!X) Y)
        (B_L_I () x X Y))
  B_L_I X1 x X2 Y)
        (APPEND X1 X2 X3)
        (IF (IMPLICADA x X3)
            ((IF (EQ X2 (x1!X4))
                 ((B_L_I X1 x1 X4 Y))
                 ((APPEND X  X2 Y))
             ))
            ((IF (EQ X2 (x1!X4))
                 ((APPEND X1 (x) X5)
                  (B_L_I X5 x1 X4 Y))
                 ((APPEND X1 (x!X2) Y))
             ))
        )
)
((IMPLICADA (x y) Z)
        (ON (x1 x2) Z)
        (SUBSET x1 x)
        (SUBSET x2 y)
)

((COMENTARIO Halla valores de hipótesis < hechos> que se desprenden
             del target y déjalos en Z5
                X es el target de entrada, Y es el target complementar
                X1,Y1 son los respectivos targets de salida))
((DEDUCE_VALORES (X Y) (X1 Y1) Z5)
        (APPEND X Y Z)
        (VARIABLES_IZQ_DE Z Z1 Z2)
        (GENERA_VALORES Z1 Z2 Z5)
        (SUBS_VALS_T Z5 X X1)
        (SUBS_VALS_T Z5 Y Y1)
)
```

```
((GENERA_VALORES Z1 Z2 Y)
        (ISALL Y1 (y1 1)
                (ON y1 Z1)
        )
        (ISALL Y2 (y2 0)
                (ON y2 Z2)
        )
        (APPEND Y1 Y2 Y)
)
((VARIABLES_IZQ_DER (()) () () ))
((VARIABLES_IZQ_DER ((y1 y2)) y1 y2))
((VARIABLES_IZQ_DER ((y1 y2) (y3 y4)|Y) Z1 Z2)
        (VARIABLES_IZQ_DER ((y3 y4)|Y) Z3 Z4)
        (INTERSEC y1 Z3 Z1)
        (INTERSEC y2 Z4 Z2)
)

((COMENTARIO SUBST ; YE VALORES EN TARGET))
((SUBS_VALS_TARGET X (Y1 Z1) (Y2 Z2))
        (SUBS_VALS_T X Y1 Y2)
        (SUBS_VALS_T X Z1 Z2)
)
((SUBS_VALS_T X () () ))
((SUBS_VALS_T X (z|Z) (z1|Z1))
        (SUBS_VALS_L X z z1)
        (SUBS_VALS_ X Z Z1)
)
((SUBS_VALS_L X (Y Z) (Y1 Z1))
        (SUBS_VALS X Y Y3)(QUITA_x 1 Y3 Y1)
        (SUBS_VALS X Z Z3)(QUITA_x 0  Z3 Z1)
)
((SUBS_VALS () Y Y))
((SUBS_VALS (x|X) Y Y1)
        (SUBS_VAL x Y Y2)
        (SUBS_VALS X Y2 Y1)
)
((SUBS_VAL (x y) Y Y1)
        (IF  (APPEND Z (x|Z1)
             ((APPEND (y|Z) Z1 Y1))
             ((EQ Y Y1))
        )
)
((COMENTARIO QUITA ELEMENTO x DE LISTA, TODAS SUS OCURRENCIAS))
((QUITA_x x () () ))
((QUITA_x x (x|X) Y)
        (QUITA_x x  X Y)
)
((QUITA_x x (y|X) (y|Y))
        (NOT EQ x y)(QUITA_x x  X Y)
)
```

```
((COMENTARIO Z es la intersección de X , Y
                se excluyen 0 y 1))
((INTERSEC X Y Z)
        (ISALL Z x (ON x X) (NOT ON x (0 1)) (ON x Y))
)

((CONMUTA_PARES () () ))
((CONMUTA_PARES ((x1 x2)!X) ((x2 x1)!X1) )
        (CONMUTA_PARES X X1)
)

((COMENTARIO primer argumento subconjunto del segundo))
((SUBSET () Y))
((SUBSET (x!X) Y)
        (ON x Y)
        (SUBSET X Y)
)

((COMENTARIO Z es la unión de los conjuntos X , Y))
((UNION X Y Z)
        (ISALL Y1 y1
                (ON y1 Y)
                (NOT ON y1 X)
        )
        (APPEND X Y1 Z)
)
((APPEND () X X))
((APPEND (x!X) Z (x!X1))
        (APPEND X Z X1)
)
((ON x (x!X) ))
((ON x (y!Y) )
        (ON x Y)
)
```

```
((COMENTARIO AL3_TRAD.LOG          TRADUCTOR DE CODIGO FUENTE
                                   A CODIGO INTERMEDIO DE LA BASE))

((COMENTARIO TRADUCE METODOS))
((TRADUCE_METODOS)
        (HALLA_METODOS X1 X2)
        (FORALL ((OR ((ON x X1)) ((ON x X2))) )
                ((TRADUCE_MET x))
        )
        (ADDCL ((METODOS_SIMPLES X1)) )
        (ADDCL ((METODOS_DIFICILES X2)) )
)

((COMENTARIO HALLA METODOS))
((HALLA_METODOS X1 X2)
        (ISALL X (x y) (METODO x y|Y))
        (ISALL X1 x     (ON (x FACIL) X))
        (ISALL X2 x     (ON (x DIFICIL) X))
)
((COMENTARIO PROCESA METODOS))
((TRADUCE_MET x)
        (METODO x y Z1 (X1 Y1))
        (DELCL ((METODO x|X)) )
        (TRAD_H Z1 Z2 Z7)
        (TRAD_H X1 (X2 X3) X7)
        (APPEND X7 Z7 X9)
        (TRADUCE_FUN X9 Y1 Y2)
        (ADDCL ((METODO x Z2 (X2 X3 Y2) )) )
)
((COMENTARIO TRAD HIPOTESIS,HECHOS))
((TRAD_H X1 (X2 X3) Z)
        (ISALL X2 (y x9) (ON (y = x9) X1) (CTE_LOG x9) )
        (ISALL Z  (y x9) (ON (y : x9) X1) )
        (ISALL X3 y (ON (x9 y) Z) )
)
((TRADUCE_FUN X Y1 Y2)
        (TRADUCCIO_TARG (~ Y1) Y3)
        (SUBS_VALS_T X Y3 Y2)
)

((TRANSF_IMP_EQ (~ X) (~ X1))
        (TRANSF_IMP_EQ X X1))
((TRANSF_IMP_EQ (X + Y) (X1 + Y1))
        (TRANSF_IMP_EQ X X1)
        (TRANSF_IMP_EQ Y Y1))
((TRANSF_IMP_EQ (X & Y) (X1 & Y1))
        (TRANSF_IMP_EQ X X1)
        (TRANSF_IMP_EQ Y Y1))
((TRANSF_IMP_EQ (X > Y) ((~X1) + Y1))
        (TRANSF_IMP_EQ X X1)
        (TRANSF_IMP_EQ Y Y1))
((TRANSF_IMP_EQ (X = Y) (X1 & Y1))
        (TRANSF_IMP_EQ (X > Y) X1)
        (TRANSF_IMP_EQ (Y > X) Y1))
((TRANSF_IMP_EQ X X))
```

```
((INTROD_NEG (~ (X & Y)) (X1 + Y1))
        (INTROD_NEG (~ X) X1)
        (INTROD_NEG (~ Y) Y1))
((INTROD_NEG (~ (X + Y)) (X1 & Y1))
        (INTROD_NEG (~ X) X1)
        (INTROD_NEG (~ Y) Y1))
((INTROD_NEG (~ (~ X)) X))
((INTROD_NEG X X))

((FNC (X & Y) (X1 & Y1))
        (FNC X X1)
        (FNC Y Y1))
((FNC (X + Y) X1)
        (FNC X X2)
        (FNC Y Y2)
        (CFNC (X2 + Y2) X1))
((FNC X X))

((CFNC ((X & Y) + Z) (X1 & Y1))
        (FNC (X + Z) X1)
        (FNC (Y + Z) Y1))
((CFNC (Z + (X & Y)) (X1 & Y1))
        (FNC (Z + X) X1)
        (FNC (Z + Y) Y1))
((CFNC X X))

((FORMA_TARGET (X & Y) Z)
        (FORMA_TARGET X X1)
        (FORMA_TARGET Y Y1)
        (APPEND X1 Y1 Z))
((FORMA_TARGET X (Y))
        (FORMA_LINEA X Y)
)
((FORMA_LINEA (X + Y) (X3 Y3))
        (FORMA_LINEA X (X1 Y1))
        (FORMA_LINEA Y (X2 Y2))
        (APPEND X1 X2 X3)
        (APPEND Y1 Y2 Y3)
)
((FORMA_LINEA (~ X) ( (X) () ) ))
((FORMA_LINEA X (() (X) ) ))

((TRADUCCION_TARG X Y)
        (TRANSF_IMP_EQ  X X1)
        (INTROD_NEG X1 X2)
        (FNC X2 X3)
        (FORMA_TARGET X3 Y7)
        (SIMPLIFICA_TARG Y7 Y)/
)

((CTE_LOG 0))
((CTE_LOG 1))
```

```
((COMENTARIO MODULO  DEPENDIENTE DEL PROBLEMA))
((COMENTARIO LA POSICION LA REPRESENTAMOS COMO UNA LISTA DE
             4 ELEMENTOS DONDE
                PRIMER ELEMENTO          DE QUIEN ES EL TURNO
                                         BLANCAS O NEGRAS
                SEGUNDO ELEMENTO         COORDENADAS REY NEGRO
                TERCER  ELEMENTO         COORDENADAS REY BLANCO
                CUARTO ELEMENTO          COORDENADAS PEON))
((CASILLAS_CRITICAS_T (X Y (7 2) (X 1) ) ))
((CASILLAS_CRITICAS_T (X Y (7 7) (X 8) ) ))
((PATRON_BP6 (BLANCAS Z (6 y1) (6 y2) _))
        (NOT EQ Z (8 y1))
        (OR ((SUM y1 1 y2))
            ((SUM y2 1 y1))
        )
)
((CASILLAS_CRITICAS (Z9 (x3 z3) (x2 z2) Y ))
        (DIST_REL (x2 z2) Y Z)
        (ON Z ((2 -1) (2 0) (2 1)))
)
((CASILLAS_CRITICAS (Z9 (x3 z3) (x2 z2) Y ))
            (DIST_REL (x2 z2) Y Z)
            (ON Z ((1 -1) (1 0) (1 1)))
            (LESS 5 x2)
)
((PEON_R7 (Z9 X Y (7 x) ) ))
((CORRE_PEON  (BLANCAS Y X X1))
        (RECTANGULO X1 Z)
        (NOT EN_RECT Y Z)
)
((CORRE_PEON  (NEGRAS Y X (x2 x3)))
        (SUM x4 1 x2)
        (RECTANGULO (x4 x3) Z)
        (NOT EN_RECT Y Z)
)
((PIERDE_PEON  (BLANCAS (y1 y2) (x1 z2) (y1 z2)) )
        (SUCC y2 z2)
        (SUM y1 2 y5) (MIN y5 8 y6)
        (SUM z2 2 z5) (MIN z5 8 z6)
        (SUM z7 2 y1) (MAX z7 1 z8)
        (NOT EN_RECT (x1 x2) ((z8 z2) (y6 z6)) )
)
((PIERDE_PEON  (BLANCAS (y1 y2) (x1 x2) (y1 z2)) )
        (SUC  z2 y2)
        (SU  y4 2 y1) (MAX 1 y4 y5)
        (SU  z4 2 z  (MAX 1 z4 z5)
        (SU  y1 2 y  (MIN y7 8 y8)
        (NO  EN_REC  (x1 x2) ((y4 z5) (y8 z2)) )
)
```

```
((PIERDE_PEON  (BLANCAS (y1 y2) (x1 x2) (z1 y2)) )
        (SUCC z1 y1)
        (SUM z4 2 z1) (MAX 1 z4 z5)
        (SUM y4 2 y2) (MAX 1 y4 y5)
        (SUM y2 2 y7) (MIN 8 y7 y8)
        (NOT EN_RECT (x1 x2) ((z5 y5) (z1 y8)) )
)
((COMENTARIO PEON A SALVO))
((PIERDE_PEON   (NEGRAS (x1 y1) (x2 y2) (x3 y3)))
        (MOVLEGAL (NEGRA  (x1 y1) (x2 y2) (x3 y3))
                  (BLANCAS (x3 y3) (x2 y2) (x3 y3))      ))
((PEON_A_SALVO  Z2)
        (NOT PIERDE_PEON Z2)

 (PEON_BLOQUEADO (Y (x1 y1) Z (x2 y1)))
        (OR ((SUM x2 1 x1)) ((SUM x2 2 x1)))
        (LESS x2 7)
)
((ENJAQUE  (NEGRAS (x1 y1) Z  (x2 y3)))
        (SUCC x3 x1)
        (OR ((SUCC y3 y1)) ((SUCC y1 y3))) )
((NOAHOGADO  x1)
             (MOVLEGAL x1 x2)     )

((DIST_R ((x1 y1) (x2 y2)) z)
        (SUM x1 z3 x2)
    -   (ABS z3 z4)
        (SUM y1 z5 y2)
        (ABS z5 z6)
        (MAX z4 z6 z)
)
((COMENTARIO DISTANCIA EN MOVS_REY EN P1 =< P2))
((NO_MAS_LEJOS X Y)
        (OR ((DIST_R X 1))
            ((NOT DIST_R X 1)(DIST_R X x) (DIST_R Y y)(NOT LESS x y))
        )
)
((COMENTARIO ES PEON DE TORRE))
((PEON_T (X1 X2 X3 (x y)))
        (OR ((EQ y 1)) ((EQ y 8)))
)
((COMENTARIO MOVIMIENTOS RESTRINGIDOS))
((MREYB  (X1 (x1 y1) Y1 (x2 y2))
         (X2 (x1 y1) Y2 (x2 y2)) ))
((RB_CERCA (X1 X2 X3 X4)  (Y1 X2 Y3 Y4))
        (NO_MAS_LEJOS (X3 X4) (Y3 Y4))
)
((RN_CERCA (X1 X2 X3 X4) (Y1 Y2 X3 Y4))
        (NO_MAS_LEJOS (X2 X4) (Y2 Y4))
)
```

```
((COMENTARIO REGLAS DE MOVIMIENTO DE FIGURAS))
((MOVLEGAL (NEGRAS  (x1 y1) (x2 y2) (x3 y3))
           (BLANCAS (X1 Y1) (x2 y2) (x3 y3)))
                (MOVREY x1 y1 X1 Y1)
                (NOT ENJAQUE (NEGRAS (X1 Y1) (x2 y2) (x3
                (NOT MOVREY X1 Y1 x2 y2)          )
((MOVLEGAL (BLANCAS (x1 y1) (x2 y2) (x3 y3))
           (NEGRAS  (x1 y1) (x2 y2) (X3 Y3)))
                (MOVPEON x3 y3 X3 Y3)
                (NOT EQ (X3 Y3) (x2 y2)))
((MOVLEGAL (BLANCAS (x1 y1) (x2 y2) (x3 y3))
           (NEGRAS  (x1 y1) (X2 Y2) (x3 y3)))
                (MOVREY x2 y2 X2 Y2)
                (NOT EQ (X2 Y2) ( x3 y3))
                (NOT MOVREY X2 Y2 x1 y1)          )

((COMENTARIO MOVIMIENTO DE REY IDEALIZADO))
((MOVREY x1 y1 x2 y2)
        (OR ((SUCC x1 x2))
            ((PRED x1 x2)))
        (OR ((SUCC y1 y2))
            ((PRED y1 y2))) )
((MOVREY x1 y1 x1 y2)
        (OR ((SUCC y1 y2))
            ((PRED y1 y2)) ) )
((MOVREY x1 y1 x2 y1)
        (OR ((SUCC x1 x2))
            ((PRED x1 x2))) )
((MOVPEON x1 y1 x2 y1)
        (SUCC x1 x2))
((SUCC x y)
        (LESS x 8)(SUM x 1 y))
((PRED x y)
        (LESS 1 x)(SUM y 1 x))

((RECTANGULO (x y) ((x y1) (0 y2)) )
        (SUM x x1 8)
        (SUM y x1 z1)
        (MIN z1 8 y2)
        (SUM x1 z2 y)
        (MAX z2 1 y1)
)
((DIST_REL (x1 y1) (x2 y2) (x3 y3))
        (SUM x2 x3 x1)
        (SUM y2 y3 y1)
)
((EN_RECT (x y) ((x1 y1) (x2 y2)) )
        (DENTRO x (x1 x2))
        (DENTRO y (y1 y2))
)
((DENTRO x (x x2) ))
((DENTRO x (x1 x) ))
```

```
((DENTRO x (x1 x2))
        (LESS x x2)
        (LESS x1 x)
)
((MAX x x x))
((MAX x x1 x)
        (LESS x1 x)
)
((MAX x x1 x1)
        (LESS x x1)
)
((MIN x x x))
((MIN x x1 x)
        (LESS x x1)
)
((MIN x x1 x1)
        (LESS x1 x)
)
((DIST x1 x2 x3)
        (SUM x2 x4 x1)
        (ABS x4 x3)
)
((ABS X Y)
        (SIGN X Z) (TIMES X Z Y))
((CIERTO))
((CIERTO X Y))
((CTELOG X) (OR ((EQ X 0)) ((EQ X 1)) ) )
```

```
((COMENTARIO ENTRADA Y SALID. ))
((COLOCA (x!X) 1 y (y!X/))
((COLOCA (x!X) z y (x!X1))
        (LESS 1 z)
        (SUM z1 1 z)
        (COLOCA X z1 y Y1)/)
((PUTMAT X (y1 y2) z X1)
        (EXTRAE X y1 X2)
        (COLOCA X2 y2 z X3)
        (COLOCA X y1 X3 X1))
((PRINTMAT (X))
        (PP X))
((PRINTMAT (x!X))
        (PRINTMAT X)
        (PP x))
((EXTRAE (x!X) 1 x))
((EXTRAE (x!X) y x1)
        (LESS 1 y)
        (NOT EQ X ())
        (SUM y1 1 y)
        (EXTRAE X y1 x1)/)
((PRINTPOS (X Y1 Y2 Y3))
        (EQ X5 ((. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)))
        (PUTMAT X5 Y1 N X6)
        (PUTMAT X6 Y2 B X7)
        (PUTMAT X7 Y3 P X8)
        (PRINTMAT X8)
        (PP))
((INICIAL_POS (BLANCAS X1 X2 X3))
        (EQ Y1 ((. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)
                (. . . . . .)))
        (PRINTMAT Y1)/
        (PP UBICACION DE REY NEGRO)
        (R X1)
        (PUTMAT Y1 X1 N Y2)
        (PRINTMAT Y2)
        (PP UBICACION DE REY BLANCO)
        (R X2)
        (PUTMAT Y2 X2 B Y3)
        (PRINTMAT Y3)
        (PP UBICACION DE PEON BLANCO)
        (R X3)
```

```
        (PUTMAT Y3 X3 P Y4)
        (PRINTMAT Y4))
((REPORTA_JUSTIF () ))
((REPORTA_JUSTIF (x!X))
        (REPORTA_CASO x)
        (REPORTA_JUSTIF X)
)
((REPORTA_CASO (X (O)) )
        (VALOR_L O Y1)
        (P "EL CONSEJO  " X "RESULTO  " Y1)(PP)
)
((REPORTA_CASO (X (Y!Z)) )
        (VALOR_L Y Y1)
        (P "EL CONSEJO   " X " RESULTO " Y1)(PP)
        (P "EL CUAL, SE SATISFACE JUGANDO") (PP)
        (PRINTPOS Z)
)
((REPORTA_CASO (X Y))
        (VALOR_L Y Y1)
        (P X "  RESULTO " Y1)(PP)
)
((VALOR_L 1 CIERTO))
((VALOR_L O FALSO )
```

Bibliografía consultada :

BERLINER 1984.

Hans J. Berliner (1984). Search vs. Knowledge: An analysis from the domain of games. In ARTIFICIAL AND HUMAN INTELLIGENCE (A. Elithorn and R. Banerji, editors) Pg 105-117.

BRATKO 78 .

Bratko, I. (1978). Proving correctness of strategies in the AL/1 assertional language. INFORMATION PROCESSING LETTERS, 7. pg 223-230.

BRATKO 81 .

Bratko, I. (1981) Knowledge-based problem-solving in AL3. Machine Intelligence 10. Ed. Michie and Pao. Ellis, Horwell and Wiley press.

BRATKO 84 .

Bratko, I. (1984) Advice and planning in chess endgames. Artificial and Human intelligence. ED. Elithorn,Alick and Banerji , Ranan.

BRATKO y KOPEC 78 .

Bratko, I., Kopec, D. & Michie, D. (1978). Pattern-based representations of chess end-game knowledge. COMPUTER JOURNAL 21, 149-153.

BRATKO y NIBLETT 79 .

Bratko, I., & Niblett, T., (1979). Conjectures and refutations in a framework for chess endgame knowledge, in EXPERT SYSTEM IN THE MICROELECTRONIC AGE, 83-102, (ed. Michie, D.).Edinburgh University Press.

CHAPA 84 .

Sergio V. Chapa Vergara (1984) Arquitectura de sistemas expertos. INFORME TECNICO No 14. Serie : Amarilla . Centro De Investigación y de Estudios Avanzados del IPN. Departamento de Ingeniería Eléctrica.

CISNEROS Y MCINTOSH 86.

Gerardo Cisneros y Harold V. McIntosh (1986). Notas sobre los lenguajes REC y Convert. Departamento de aplicación de Microcomputadoras. Institoto de Ciencias. U.A.P.

CROSSON 70 .

Frederick J. Crosson (1970) Inteligencia Humana e Inteligencia Artificial . Fondo de Cultura Económica.

FREY y ATKIN 78 .

Peter W. Frey y Larry R Atkin (1978) Creating a Chess Player en BYTE de Octubre a Diciembre 1978.

HOROWITZ 76 .

Ellis Horowitz Sartaj Sahni (1976) . Computer Science Press, Inc. Computer Software Engineering Series.

HAYES-ROTH y WATERMAN 83 .

Hayes Roth, Frederick y Waterman, Donald A. (1983) , Building Expert System , Addison-Wesley Publishing Company, Massachusetts.

HILGARD y BOWER 1984 .

Ernest R. Hilgard y Gordon H. Bower (1984). Teorías del aprendizaje. Ed. Trillas.

KOTOV 71 .

Alexander Kotov (1971) Think like a Grandmaster. Chess Digest Inc. Dallas Texas.

KOTOV 83 .

Alexander Kotov (1983) Juegue como un gran maestro

NEWBORN 75 .

Monroe Newborn (1975) Computer Chess . ACM Monograph Series.

PITRAT 76 .

Pitrat, J., (1976). A program for learning to play chess. in Chen, ed., Pattern Recognition and artificial intelligence (Academic Press, New York, 1976) 399-419.

PITRAT 77 .

Pitrat, J., (1977). A chess combinations program which uses plans. Artificial Intelligence 8. 275-321.

QUINLAN 79 .

Quinlan,J.R., (1979) . Discovering rules by induction from large collections of examples. Expert Systems in the Micro Electronic Age, pp. 168-201. (ed. Michie, D.). Edinburgh: Edinburgh University Press.

MAZIELIS 3, 6 .

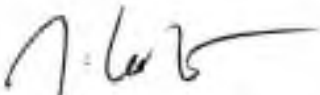
Mazieliz, I. Finales de peones , pag. 3 - 6 .

Ed. Martinez Roca, Col. Escaqes.

ZDRAHAL 1987

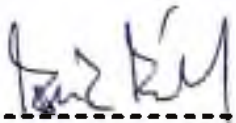
Zdenek Zdrahal (1987). Adquisición de conocimiento y aprendizaje inductivo. En "PRIMER CURSO INTERNACIONAL DE SISTEMAS EXPERTOS". I.P.N. CINVESTAV. Pag 127-148.

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de - Investigación y de Estudios Avanzados de Instituto - Politécnico Nacional, aprobó esta tesis el 28 de abril de 1988.



Dr. Josef Kolar

Dr. Guillermo Morales Luna.



Dr. Zdeněk Zdráhal Horová.