Representación del Conocimiento y el Razonamiento

Implementación de un generador de contenido web semántico con posiciones didácticas de ajedrez

MEMORIA FINAL

Estudiante: José Luis Herreros Lucas

Director: David Isern Alarcón

Ingeniería Informática

Junio 2011

DEDICATORIA

Este trabajo se lo dedico a mis hijos, Jorge Alejandro y Jaime Adrián.

Por ellos y para ellos.

AGRADECIMIENTOS

Como indica el plan docente "*El Proyecto fin de carrera (PFC) es una asignatura que está pensada para realizar un trabajo de síntesis de los conocimientos adquiridos en otras asignaturas de la carrera y que requiera ponerles en práctica conjuntamente en un trabajo concreto*". Y así he intentado que sea gracias a los contenidos de muchas de las asignaturas cursadas, como comentaré a continuación.

El conocimiento adquirido en la asignatura *Ingeniería del software orientada al objeto* me ha permitido definir adecuadamente las clases, propiedades e individuos contenidos en la base de conocimiento que ha sido generada. Los conocimientos adquiridos en la asignatura *Metodología y gestión de proyectos informáticos* me han permito realizar una planificación de proyecto real, y bien ajustada a los requerimientos y tiempos del proyecto. Cursar las asignaturas de *Compiladores I y II* ha sido vital para el desarrollo de este proyecto, dado que la estructura del proceso desarrollado se ajusta perfectamente con las prácticas realizadas en esta asignatura. El material de la asignatura de *Proceso de ingeniería del software* me ha permitido dar un enfoque metódico a la hora de definir la ontología presentada en esta memoria. También, lo aprendido en la asignatura de *Interacción humana con los ordenadores* me ha permitido concebir el diseño de la página web teniendo también en cuenta características de accesibilidad. Los contenidos de las asignaturas de *Inteligencia Artificial I y II* me han servido para entender – es más, disfrutar – todas las materias relacionadas con este área, principal motivo por el que elegí un área de desarrollo del PFC relacionada con estas asignaturas. Por último, la asignatura de *Competencia comunicativa para profesionales de informática* me ha permitido – eso espero – estructurar los contenidos de esta memoria de una forma más acorde con el tipo de texto presentado. En general, muchas otras asignaturas también me han servido para mejorar mis conocimientos del lenguaje de desarrollo Java y HTML -básicos cuando inicié la Ingeniería-, y de gran utilidad a la hora de afrontar la fase de diseño e implementación de este PFC. Por todo ello, agradezco al equipo de consultores que me han ayudado todos estos años por su empeño, paciencia y dedicación. Muchas gracias a todos.

Haberme permitido realizar un PFC de un área tan interesante como la representación de conocimiento para un contenido del que podría decirse que soy un gran aficionado, el ajedrez, ha sido una verdadera satisfacción y, por ello, tengo que agradecer esta gran oportunidad al Dto. de Inteligencia Artificial y, en especial, a David Isern que apostó por la realización específica de este proyecto. Como él sabe, partimos de varias ideas relacionadas con el área de IA y gracias a su estimable ayuda pudimos concretar un objetivo real y realizable en el corto plazo de tiempo establecido. David, muchas gracias.

Por último, este proyecto no podría haberlo realizado si las personas más cercanas no me hubiesen ayudado. Compaginar el trabajo, los estudios y el cuidado de la familia no hubiese sido posible sin la ayuda y apoyo de mi mujer, Conchi. Por ello, mi eterna gratitud y reconocimiento. Siempre estaré en deuda con ella.

Espero, además, que el contenido de este trabajo pueda ser de utilidad para muchas personas. Por mi parte, ya he puesto en práctica el uso de las páginas web generadas desde el mismo día en que entregué la PEC3, con el producto final desarrollado. Espero que pueda servir para mejorar el puesto en el que quedará mi hijo en el Campeonato de España sub-10 que se celebrará dentro de pocos días, y que pueda igualmente mejorar el nivel de juego de cualquier otro jugador de este "deporte-ciencia". Es probable que lo próximo que realice, dado mi interés en esta materia, sea la explotación de la base de conocimiento generada desde buscadores web semánticos, objetivo no incluido en este trabajo pero que sí podría ser el objetivo de cualquier otro proyecto en la UOC.

Estoy a disposición de cualquier persona interesada en este tema para comentar mis futuros avances en esta materia, o para comentar vuestros avances.

"The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users. Such an agent coming to the clinic's Web page will know not just that the page has keywords such as "treatment, medicine, physical, therapy" (as might be encoded today) but also that Dr. Hartman works at this clinic on Mondays, Wednesdays and Fridays and that the script takes a date range in yyyy-mm-dd format and returns appointment times.

[…]

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present."

"La Web Semántica proporcionará estructura para el contenido significativo de páginas web, creando un entorno donde los agentes de software que viajan de página a página fácilmente puede llevar a cabo tareas sofisticadas para los usuarios. Así, un agente que venga de la página web de la clínica no solo sabrá que la página tiene palabras clave como "tratamiento, medicina, física, terapia" (como podría ser codificado hoy), sino también que el Dr. Hartman trabaja en esta clínica los lunes, miércoles y los viernes y que el script toma un intervalo de fechas en aaaa-mm-dd y devuelve la cita.

[…]

La web semántica no es una web independiente, sino una extensión de la actual, en la que la información se le da un significado bien definido, facilitando a computadoras y personas el trabajar en cooperación. Los primeros pasos para desarrollar la Web Semántica en la estructura de la actual Web ya están en marcha. En un futuro próximo, esta evolución marcará el comienzo de una nueva funcionalidad importante como las máquinas son mucho más capaces de procesar y "entender" los datos que se limitan a mostrar en la actualidad."

Tim Berners-Lee, James Hendler y Ora Lassila

Scientific American: Feature Article: The Semantic Web: May 2001

http://old.hki.uni-koeln.de/temp/SemWebSeminal.pdf

ÍNDICE DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

0 DEFINICIÓN GENERAL DEL PROYECTO

Al inicio de esta memoria hemos visto un par de párrafos escritos por Tim Berners-Lee hace ya más de 10 años. A continuación, presentamos una aplicación práctica de la concepción semántica de la web que tuvo este gran visionario.

0.1 Objetivos

Tanto los monitores de ajedrez de niños que se inician en este deporte como los grandes maestros, dedican gran parte de su tiempo a preparar y/o a analizar posiciones que permitan mejorar sus capacidades de juego, o las capacidades de sus alumnos. Estas posiciones son didácticas dado que podemos aprender de ellas los múltiples secretos de este fascinante "juego-ciencia", como lo determinan algunos escritores y maestros. Estos ejercicios están basados en temas relacionados con la estrategia y la táctica, así como en posiciones determinadas dentro de una partida (apertura, medio juego o finales). Desgraciadamente, es complicado obtener buenos ejercicios, a no ser que adquiramos libros o bases de datos especializadas (con coste) con las correspondientes posiciones. Si hablamos del mundo web, existen páginas interesantes que permiten realizar ejercicios on-line, pero estas páginas son "estáticas", en el sentido de que el usuario no puede realizar una selección "inteligente" de dichos ejercicios y/o posiciones.

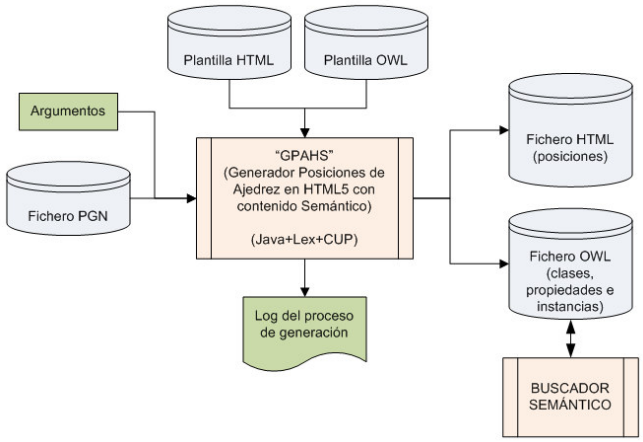
Las posibilidades de la web semántica podrían reducir en gran medida el tiempo invertido por estos profesionales a la hora de seleccionar y/o buscar los mejores ejercicios en cada momento.

El objetivo del presente Proyecto Final de Carrera (PFC en adelante) trata de dar un primer paso a la hora de resolver la problemática anteriormente mencionada, realizando el análisis, diseño e implementación de un Generador de Posiciones de Ajedrez en formato HTML5 y contenido Semántico (**GPAHS** en adelante), que permita crear posiciones determinadas de una partida de ajedrez real o ficticia, a modo de ejercicios con resolución, todo ello a partir de ficheros con formato "Portable Game Notation" (PGN en adelante), muy utilizados por este tipo de profesionales.

A modo de resumen (ya veremos el detalle en el siguiente capítulo), el aplicativo desarrollado en Java procesará los ficheros de entrada en formato PGN, con el apoyo de las herramientas JLex/CUP, para generar:

- Páginas HTML5 que contengan diagramas de posiciones de ajedrez, utilizando las nuevas características "canvas" del lenguaje. También se incluye la solución propuesta para el ejercicio, así como otras características de la posición: jugadores, torneo, ciudad, fecha del encuentro, etc.
- Contenido web semántico. Base de conocimiento con las clases, propiedad e instancias de la ontología, que serán igualmente definidas durante el desarrollo del PFC.

Gráficamente, la implementación a desarrollar será la siguiente:



Una vez generadas las páginas web, se podrían publicar sobre un servidor para que los buscadores semánticos actuales que tuviesen indexada nuestra ontología (base de conocimiento generada) puedan realizar búsquedas "inteligentes" en dichas páginas.

NOTA: *El desarrollo del buscador semántico, o cualquier acción sobre los buscadores semánticos ya existentes en el mercado, no forman parte de los objetivos de este PFC.*

Figura I: Generador GPAHS, esquema general.

Algunos de los puntos más importantes a analizar para alcanzar este objetivo son los siguientes:

- Analizar el formato normalizado de los ficheros PGN para desarrollar un analizador léxico / sintáctico con las herramientas JLex/CUP que permita tratar su contenido.
- Analizar el formato FEN, el cual refleja una posición exacta de una partida de ajedrez, así como toda la información que la complementa: turno de juego (blancas o negras), enroques posibles, movimientos sin captura, etc.
- Analizar y documentar la sintaxis del lenguaje ontológico OWL (Ontology Web Language), un lenguaje de marcado para publicar y compartir datos usando ontologías en la web. Probablemente, el análisis más importante para el desarrollo de este PFC.
- Adquirir el conocimiento necesario de manejo de la herramienta Protégé (editor ontológico) para definir la ontología requerida, a partir de la información que podremos obtener de los ficheros PGN, y de la documentación relacionada con este formato.
- Analizar cómo generar contenido semántico (instancias) a incluir a una base de conocimiento. Dicho contenido será igualmente generado por el programa **GPAHS**.
- Analizar cómo incluir contenido gráfico mediante las nuevas características "canvas" de HTML5. Dicho contenido será incluido automáticamente por el programa **GPAHS,** a partir de la información contenida en una plantilla HTML.
- Analizar cómo incluir el resto de contenido (soluciones, jugadores, etc.) utilizando igualmente HTML5. Dicho contenido, como en todos los casos anteriores, será incluido automáticamente por el programa **GPAHS**.

0.2 Entregables previstos

Los entregables previstos en este PFC son los siguientes:

- Un primer plan de trabajo, que recoge la planificación y estimación de las tareas necesarias para llevar a cabo los objetivos previstos.
- La definición de la ontología relacionada con posiciones de ajedrez, realizada con la herramienta Protégé.
- Los desarrollos Java/JLex/CUP que conforman el generador de código HTML5.
- La memoria, que es el documento que sintetiza el trabajo realizado y que presentará los objetivos propuestos en este plan de trabajo. Incorporará toda la información relevante para comprender el problema planteado, la metodología utilizada para su resolución, así como el detalle de la solución entregada. Esta memoria será formada a partir de los entregables parciales del PFC:
  - Entregable PEC1: Descripción de objetivos, entregables y plan de proyecto.
  - Entregable PEC2: Análisis y diseño de la aplicación.
  - Entregable PEC3: Desarrollo y pruebas de la aplicación desarrollada.
- La presentación final, que resumirá de forma clara y concisa el trabajo realizado, así como los resultados obtenidos en el PFC.

0.3 Marco temporal

El plazo establecido para la entrega de todos los entregables resultantes del PFC (producto, memoria y presentación) es el 16 de junio de 2011. La planificación elaborada para este PFC, así como la estimación del esfuerzo presente en este plan de trabajo permiten desarrollar todos los objetivos definidos.

La fecha inicial del proyecto es la fecha de inicio del semestre, el 2 de marzo de 2011. El plan de fechas de entregables es el siguiente:

- Entrega del plan de trabajo (PEC1) 12 de marzo de 2011.
- Entrega análisis y diseño (PEC2) 16 de abril de 2011.
- Entrega desarrollo y pruebas (PEC3) 21 de mayo de 2011.
- Entrega memoria y presentación final 16 de junio de 2011.

NOTA: *En los siguientes apartados de este plan de trabajo se detalla el contenido de cada entregable.*

A continuación, también se incluye en modo gráfico el calendario previsto de entregas:

Leyenda:

PEC1
PEC2
PEC3
FINAL

Marzo

L	M	X	J	V	S	D
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Abril

L	M	X	J	V	S	D
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Mayo

L	M	X	J	V	S	D
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Junio

L	M	X	J	V	S	D
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Figura II: Calendario de entregas.

0.4 Ciclo de vida

Para conseguir una estimación de trabajo y una planificación lo más real posible, y teniendo en cuenta el número de entregables requeridos para esta PEC, se ha considerado como más conveniente utilizar el ciclo de vida de desarrollo de proyectos “en cascada”, con las siguientes apreciaciones:

- Las fases de análisis de requerimientos, análisis y diseño serán realizadas y entregadas en la PEC2.
- Las fases de codificación (desarrollo) y pruebas serán realizadas y entregadas en la PEC3.
- La entrega final contiene la memoria del PFC, así como la presentación final, la cual reflejará las principales características del proyecto realizado.

NOTA: *Excluimos de esta metodología las fases de implantación y mantenimiento, dado que no son requeridas en los objetivos concretos de este PFC.*

0.5 Recursos e infraestructura

En este apartado se detallan los recursos hardware y software requeridos para el desarrollo de este PFC.

0.5.1 Hardware

La previsión de recursos hardware a utilizar durante el desarrollo del proyecto son los siguientes:

- Equipo portátil Dell Latitude E6400, Windows XP en inglés, con 3 GB. de memoria y 300 GB. de disco duro.
- Teléfono Samsung Galaxy S, donde se realizarán algunas pruebas, desde su navegador, de las páginas web obtenidas por el programa **GPAHS**.

0.5.2 Software

La previsión de recursos software a utilizar durante el desarrollo del proyecto son los siguientes:

- Programa Chessbase 10: requerido para el análisis de los ficheros "pgn", a utilizar como entrada en nuestro generador.
- Para la elaboración de todos los documentos a generar en el proyecto se utilizará el paquete MS Office 2007. En concreto:
  - MS Word 2007: Elaboración de documentos "doc".
  - MS Excel 2007: Elaboración de cuadros y estimación preliminar de esfuerzos contenida en el plan de trabajo.
  - MS Project 2007: Elaboración del diagrama Gantt del plan de proyecto.
  - MS Visio 2007: Elaboración de gráficos.
  - MS Power Point 2007: Elaboración de la presentación final del PFC.
  - MS OneNote 2007: Inventario de notas y/o web consultadas, para su inclusión en cada uno de los entregables y memoria final (referencias).
- Protégé: Editor de ontologías.
- IDE Netbeans 6.8: Editor de desarrollo Java (utilizado en más de una ocasión durante la carrera).
- Navegador Safari 5.0.2: Herramienta para el desarrollo de las pruebas locales, sobre el equipo de desarrollo anteriormente mencionado.
- SnagIt 8: Capturas de pantallas (o secciones de) a incluir en los entregables del PFC.
- Dropbox, para el salvado permanente (back-ups diarios) del contenido del PFC.
- MindJet MindManager Pro 7 para el desarrollo de gráficos de carácter jerárquico (mapas mentales).
- Adobe Acrobat Writer 8.1, para la inclusión de notas de referencia en los documentos PDF que sean utilizados durante el desarrollo del PFC.

NOTA: *Entendemos que esta lista de productos software satisface todas las necesidades iniciales del proyecto. No obstante, podría surgir la necesidad de nuevos productos, una vez se analice y diseñe el 100% del generador a desarrollar. Por ejemplo, existen herramientas Poen Source de transformación de ficheros de formato "cbh" a formato "pgn", y es posible su utilización para la obtención de ficheros de entrada a nuestro aplicativo.*

0.6 Estructura del proyecto y descomposición en actividades

Como ya se ha comentado, se utilizará una metodología de desarrollo en cascada. Teniendo en cuenta las jornadas establecidas en el calendario oficial para el desarrollo de este PFC y la dedicación estimada por jornada, a continuación se detalla el conjunto de actividades previstas como contenido, así como la descripción de cada una de dichas actividades.

0.6.1 Descomposición en actividades

Las actividades que compondrán el proyecto son las siguientes:

Descomposición estructural de actividades (WBS)			
Código actividad	nombre de la actividad		
	Nivel 1	Nivel 2	Nivel 3
01	Plan de trabajo (PEC1)		
01.01		Selección PFC y enfoque	
01.02		Elaboración plan de trabajo PFC	
01.03		Entrega plan de trabajo PFC	
02	Análisis y diseño (PEC2)		
02.01		Análisis formato PGN	
02.02		Análisis formato FEN	
02.03		Análisis y diseño de uso de librerías LEX/CUP	
02.04		Análisis lenguaje OWL	
02.04.01			Introducción OWL
02.04.02			OWL y RDF/RDFS
02.04.03			Sublenguajes OWL
02.04.04			Descripción del lenguaje OWL
02.04.05			Las capas de OWL
02.05		Diseño ontología de ajedrez con Protégé	
02.05.01			Descripción herramienta Protégé
02.05.02			Diseño ontología de ajedrez
02.06		Análisis y diseño de etiquetas semánticas HTML5	
02.07		Análisis y diseño "canvas" en HTML5	
02.08		Otras características HTML requeridas	
02.09		Diseño del generador semántico HTML5	
02.10		Entrega análisis y diseño (PEC2)	
03	Desarrollo y pruebas		
03.01		Desarrollo generador: Tratamiento fichero de entrada	
03.02		Desarrollo generador: Contenido semántico	
03.03		Desarrollo generador: Figuras canvas	
03.04		Desarrollo generador: Otras características	
03.05		Pruebas integradas	
03.06		Entrega desarrollo y pruebas (PEC3)	
04	Elaboración memoria y presentación del PFC (Entrega final)		
04.01		Elaboración memoria del PFC	
04.02		Elaboración de la presentación del PFC	
04.03		Entrega final	

Tabla I: Actividades del proyecto

0.6.2 Descripción de las actividades

En esta sección incluimos las principales actividades que compondrán el proyecto.

Plan de trabajo (PEC1)

Esta fase del proyecto contempla la elección y enfoque del PFC a desarrollar, la planificación detallada del proyecto; fases, esfuerzo y calendario de cada una de las actividades. Las actividades previstas en esta fase son las siguientes:

- Selección PFC y enfoque: El punto de partida ha sido la selección del PFC a desarrollar, dentro del abanico de posibilidades propuestas, así como el enfoque a desarrollar, una vez seleccionado.
- Elaboración plan de trabajo PFC: La elaboración del plan de trabajo queda reflejada en este documento, y se corresponde con el entregable de la primera PEC del PFC.
- Entrega plan de trabajo PFC: Esta actividad contempla las acciones relacionadas con la entrega de todo el material elaborado para el desarrollo del plan de trabajo.

Análisis y diseño (PEC2)

Esta fase contempla todas las actividades relacionadas con el análisis y diseño requeridas para el desarrollo del programa **GPAHS** (Generador de Posiciones de Ajedrez en HTML5 y con contenido semántico), objetivo final del proyecto. Las actividades principales de la fase son las siguientes:

- Análisis formato PGN: El programa **GPAHS** debe leer un fichero de entrada en formato "PGN", el cual contiene cada una de las posiciones de ajedrez a tratar, punto de partida para su resolución, junto con la solución propuesta y otras características de dicha posición (jugadores, torneo, fecha, ciudad del encuentro, etc.). El formato "PGN" es un formato estándar utilizado por la mayoría de los programas de gestión de bases de datos de partidas de ajedrez. En esta actividad se analizarán las principales características de este formato, para poder tratar correctamente su contenido desde el programa **GPAHS**.
- Análisis formato FEN: Para describir una posición determinada de una partida de ajedrez existen varios métodos. Para analizar una partida completa se indican todos los movimientos realizados por ambos jugadores en una notación denominada "algebraica", aunque existen otras posibilidades. Por el contrario, cuando sólo nos interesa una posición determinada de una partida, para analizar cuál sería el siguiente mejor movimiento (ejercicio), la notación que normalmente se utiliza es la "FEN" (Forsyth-Edwards Notation). En esta actividad se analiza dicho formato, requerido para poder transformar esta información en el gráfico HTML a generar.
- Análisis y diseño de uso de librerías Lex/CUP: Estas librerías, ya utilizadas en la asignatura de compiladores, serán la base para poder interpretar el contenido del fichero PGN mediante un análisis léxico/sintáctico de su contenido. Gracias a esta interpretación se obtendrá toda la información relevante para la generación de código HTML, tanto del contenido semántico como del contenido gráfico.

- Análisis lenguaje OWL: Consideramos esta actividad la más importante dentro de la fase de análisis y diseño, por ello, la que mayor contenido tendrá. Gracias a este análisis se podrá definir correctamente la ontología relacionada con los objetivos de la PFC (diseño), así como su uso posterior en las páginas HTML5 (desarrollo).
- Diseño ontología de ajedrez con Protégé: Gracias a la herramienta Protégé, se realizará en esta actividad la definición de la ontología relacionada con los objetivos del PFC. Una partida y la información relacionada con la misma (jugadores, ELOs, torneo, fecha, tema táctico) serán los objetos principales de la ontología.
- Análisis y diseño de etiquetas semánticas HTML5: Llegamos a las actividades directamente relacionadas con el objeto a generar, páginas web escritas en lenguaje HTML5. Este lenguaje permite la inclusión de contenido semántico, y en esta actividad se analizará cómo lo realizaremos.
- Análisis y diseño "canvas" en HTML5: En esta nueva actividad se analizarán las nuevas características del lenguaje HTML, en concreto, las relacionadas con la generación de gráficos gracias a las funciones "canvas" incorporadas. Estas nuevas características nos permitirán "dibujar" la posición concreta de una partida, la cual planteará un ejercicio a resolver.
- Otras características HTML requeridas: Para completar la página HTML a generar necesitamos incluir el resto de información: todos los atributos relacionados con la partida (jugadores, quien tiene el turno de juego, torneo, fecha, etc.), así como la solución del ejercicio, cuando el usuario lo requiera. En esta actividad se analizará y se diseñará esta parte del código a generar.
- Diseño del generador semántico HTML5: Esta actividad contempla el diseño general del programa **GPAHS** a desarrollar, y las bases que determinarán la definición de los casos de uso y pruebas del programa.
- Entrega Análisis y Diseño (PEC2): Esta actividad contempla las acciones relacionadas con la entrega de todo el material elaborado en esta fase: análisis y diseño realizado.

Desarrollo y pruebas (PEC3)

Llegamos a la fase de desarrollo y pruebas de dicho desarrollo. Para poder analizar con mejor criterio el grado de avance durante su desarrollo, esta fase se ha divido en las siguientes actividades:

- Desarrollo generador: Tratamiento fichero de entrada: En esta primera actividad se desarrollará la parte del programa Java que mediante las librerías Lex/CUP permitirán realizar un análisis léxico/sintáctico del fichero de entrada, almacenando en memoria toda la información relevante para su posterior uso.
- Desarrollo generador: Contenido semántico: Esta actividad contempla las acciones de desarrollo necesarias para que a partir de la información almacenada en memoria (actividad anterior) se genere la estructura básica de la página web, junto con el contenido semántico oportuno.
- Desarrollo generador: Figuras canvas: En esta actividad se incluirá en la página web los diagramas que representan la posición de ajedrez, punto de partida del ejercicio a resolver.

- Desarrollo generador: Otras características y soluciones: En esta actividad concluimos el desarrollo, incluyendo las características adicionales del ejercicio, así como un procedimiento para que el usuario pueda ver la solución propuesta. Esta actividad incluirá todos elementos no contemplados en las actividades anteriores de esta fase.
- Pruebas integradas: Esta actividad contendrá la evidencia de todas las pruebas realizadas que reflejen el correcto funcionamiento del programa **GPAHS**. Esta actividad también contempla la búsqueda de bases de datos en formato "PGN" susceptibles de ser transformadas en páginas HTML.
- Entrega desarrollo y pruebas (PEC3): Esta actividad contempla las acciones relacionadas con la entrega de todo el material elaborado en esta fase: desarrollo y pruebas.

Elaboración memoria y presentación del PFC (Entrega final)

En esta última fase se elaborará la memoria final del PFC, así como la presentación de dicho PFC. Las actividades contempladas en esta fase son las siguientes:

- Elaboración memoria del PFC: Actividad que contempla la composición de una memoria final con el contenido ordenado de la documentación elaborada durante el desarrollo del PFC.
- Elaboración de la presentación del PFC: Actividad final en la que se realizará la presentación (formato MS Power Point 2007) con las principales características y acciones realizadas durante el proyecto.
- Entrega final: Esta actividad contempla las acciones relacionadas con la entrega de todo el material elaborado en esta fase: memoria final y presentación del PFC.

0.7 Planificación temporal del proyecto

En esta sección del plan de proyecto se incluye toda la información relacionada con el espacio temporal del mismo, así como los esfuerzos previstos y las actividades condicionadas a otras (predecesoras).

0.7.1 Calendario para el desarrollo del proyecto

Como ya se indicó en el apartado 1.3 de este documento (marco temporal), el inicio del proyecto se estable en la fecha 2 de marzo de 2011 (inicio del semestre) y termina en la fecha 16 de junio de 2011 (finalización del semestre para el PFC). El tiempo establecido en jornadas para la elaboración de cada entregable es el siguiente:

- Entrega PEC1 (plan de trabajo): 11 jornadas.
- Entrega PEC2 (análisis y diseño): 35 jornadas.
- Entrega PEC3 (desarrollo y pruebas): 35 jornadas.
- Entrega final (memoria y presentación): 26 jornadas.

Por ello, un total de 107 jornadas.

Para presentar la estimación en horas se realiza la siguiente aproximación:

- En las jornadas laborables se tendrá una dedicación de 3 horas (de 8:00 a 9:00 y de 13:00 a 15:00).
- En las jornadas no laborables (fin de semana y festivos) también se realizará una dedicación de 3 horas (8:00 a 11:00).

Con estas premisas, las horas previstas para cada entrega son las siguientes:

- Entrega PEC1 (plan de trabajo): 33 horas.
- Entrega PEC2 (análisis y diseño): 105 horas.
- Entrega PEC3 (desarrollo y pruebas): 105 horas.
- Entrega final (memoria y presentación): 78 horas.

Por ello, un total de 321 horas.

NOTA: *Como se ha indicado, se trata de una estimación. En cada jornada se realizarán todas las horas requeridas para cumplir el calendario previsto, mostrado a continuación.*

En la siguiente figura se muestra esta misma información en modo tabla:

PEC1	Día		M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Subtotal	Total
	Marzo	Jornadas		1	1	1	1	1	1	1	1	1	1	1																				11	11
		horas		3	3	3	3	3	3	3	3	3	3	3																				33	33
PEC2	Día		M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Subtotal	Total
	Marzo	Jornadas													1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	35
		horas													3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	57	105
	Día		V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30			
	Abril	Jornadas	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1																16	
		horas	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3																48	
PEC3	Día		V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		Subtotal	Total
	Abril	Jornadas																	1	1	1	1	1	1	1	1	1	1	1	1	1	1		14	35
		horas																	3	3	3	3	3	3	3	3	3	3	3	3	3	3		42	105
	Día		D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
	Mayo	Jornadas	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1											21	
		horas	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3											63	
FINAL	Día		D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Subtotal	Total
	Mayo	Jornadas																						1	1	1	1	1	1	1	1	1	1	10	26
		horas																						3	3	3	3	3	3	3	3	3	3	30	78
	Día		X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
	Junio	Jornadas	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1																16	
		horas	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3																48	

21/04/2011	**Jueves Santo**
22/04/2011	**Viernes Santo**
02/05/2011	**Lunes siguiente fiesta del trabajo**
23/05/2011	**Corpus Christi**

Figura III: Estimación de esfuerzos por entrega.

0.7.2 Planificación temporal

En esta sección realizaremos una planificación con la herramienta MS Project 2007, donde se podrá también analizar las precedencias entre las actividades.

Precedencia de las actividades

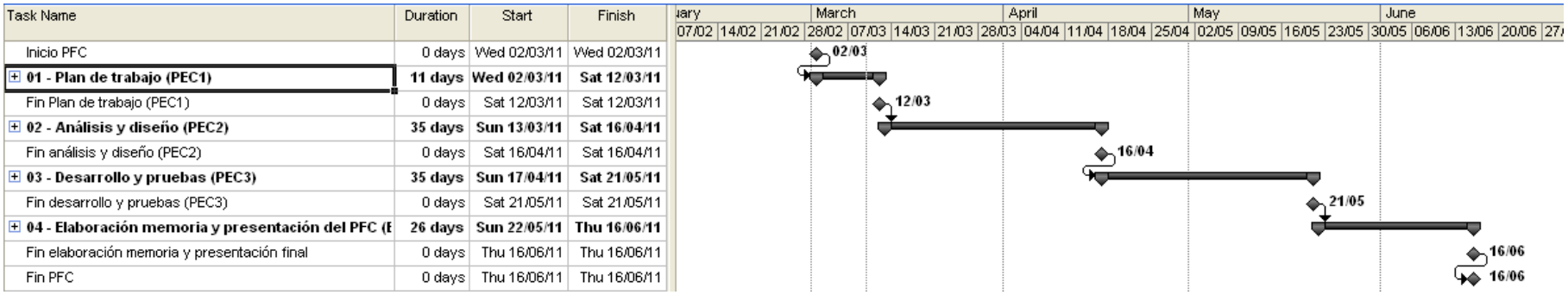
Las reglas utilizadas para determinar las precedencias, teniendo en cuenta que se empleará una metodología en cascada, son las siguientes:

- La fase de elaboración del plan de proyecto es anterior a la fase de análisis y diseño.
- La fase de análisis y diseño es anterior a la fase de desarrollo y pruebas.
- La fase de desarrollo y pruebas es anterior a la fase de elaboración de la memoria y presentación final.

Dentro de cada fase, cada una de sus actividades seguirá igualmente una secuencia en cascada, dado que el proyecto será realizado por una única persona y no se detectan necesidades de paralelismo de actividades. Por ello, no se definirá en el plan de proyecto ninguna actividad en paralelo con otra.

Planificación propuesta

Con las fases, actividades por fase y precedencias comentadas, la planificación propuesta de alto nivel es la siguiente:



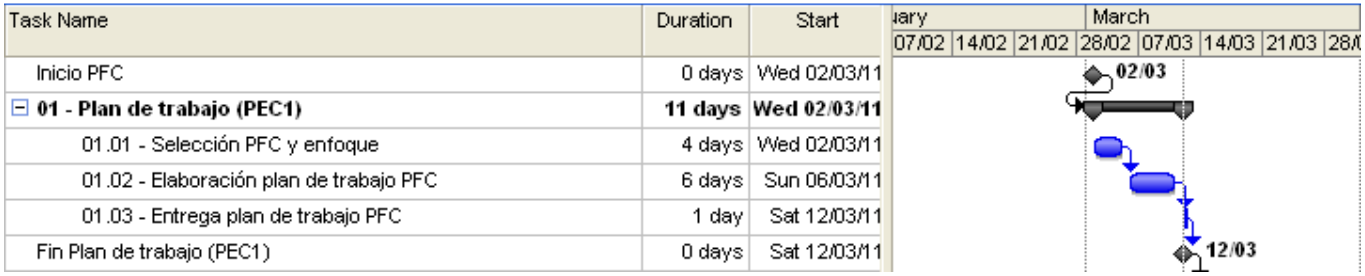
Task Name	Duration	Start	Finish
Inicio PFC	0 days	Wed 02/03/11	Wed 02/03/11
01 - Plan de trabajo (PEC1)	**11 days**	**Wed 02/03/11**	**Sat 12/03/11**
Fin Plan de trabajo (PEC1)	0 days	Sat 12/03/11	Sat 12/03/11
02 - Análisis y diseño (PEC2)	**35 days**	**Sun 13/03/11**	**Sat 16/04/11**
Fin análisis y diseño (PEC2)	0 days	Sat 16/04/11	Sat 16/04/11
03 - Desarrollo y pruebas (PEC3)	**35 days**	**Sun 17/04/11**	**Sat 21/05/11**
Fin desarrollo y pruebas (PEC3)	0 days	Sat 21/05/11	Sat 21/05/11
04 - Elaboración memoria y presentación del PFC (E	**26 days**	**Sun 22/05/11**	**Thu 16/06/11**
Fin elaboración memoria y presentación final	0 days	Thu 16/06/11	Thu 16/06/11
Fin PFC	0 days	Thu 16/06/11	Thu 16/06/11

Figura IV: Plan de proyecto, fases principales.

NOTA: *Junto al plan de proyecto (este documento) se entrega el diagrama gantt desarrollado con la herramienta MS Project 2007*.

El detalle de cada fase es el siguiente:

Fase 1, Plan de proyecto (PEC1):



Task Name	Duration	Start
Inicio PFC	0 days	Wed 02/03/11
01 - Plan de trabajo (PEC1)	**11 days**	**Wed 02/03/11**
01.01 - Selección PFC y enfoque	4 days	Wed 02/03/11
01.02 - Elaboración plan de trabajo PFC	6 days	Sun 06/03/11
01.03 - Entrega plan de trabajo PFC	1 day	Sat 12/03/11
Fin Plan de trabajo (PEC1)	0 days	Sat 12/03/11

Figura V: Plan de proyecto, PEC1.

Fase 2, análisis y desarrollo (PEC2):

Task Name	Duration	Start	Finish
02 - Análisis y diseño (PEC2)	**35 days**	**Sun 13/03/11**	**Sat 16/04/11**
02.01 - Análisis formato PGN	2 days	Sun 13/03/11	Mon 14/03/11
02.02 - Análisis formato FEN	1 day	Tue 15/03/11	Tue 15/03/11
02.03 - Análisis y diseño de uso de librerías LEX/CUF	3 days	Wed 16/03/11	Fri 18/03/11
02.04 - Análisis lenguaje OWL	**13 days**	**Sat 19/03/11**	**Thu 31/03/11**
02.04.01 - Introducción OWL	2 days	Sat 19/03/11	Sun 20/03/11
02.04.02 - OWL y RDF/RDFS	3 days	Mon 21/03/11	Wed 23/03/11
02.04.03 - Sublenguajes OWL	3 days	Thu 24/03/11	Sat 26/03/11
02.04.04 - Descripción del lenguaje OWL	3 days	Sun 27/03/11	Tue 29/03/11
02.04.05 - Las capas de OWL	2 days	Wed 30/03/11	Thu 31/03/11
02.05 - Diseño ontología de ajedrez con Protég	**6 days**	**Fri 01/04/11**	**Wed 06/04/11**
02.05.01 - Descripción herramienta Protégé	3 days	Fri 01/04/11	Sun 03/04/11
02.05.02 - Diseño ontología de ajedrez	3 days	Mon 04/04/11	Wed 06/04/11
02.06 - Análisis y diseño de etiquetas semánticas HT	2 days	Thu 07/04/11	Fri 08/04/11
02.07 - Análisis y diseño canvas en HTML5	3 days	Sat 09/04/11	Mon 11/04/11
02.08 - Otras características HTML requeridas	2 days	Tue 12/04/11	Wed 13/04/11
02.09 - Diseño del generador semántico HTML5	2 days	Thu 14/04/11	Fri 15/04/11
02.10 - Entrega análisis y diseño (PEC2)	1 day	Sat 16/04/11	Sat 16/04/11
Fin análisis y diseño (PEC2)	0 days	Sat 16/04/11	Sat 16/04/11

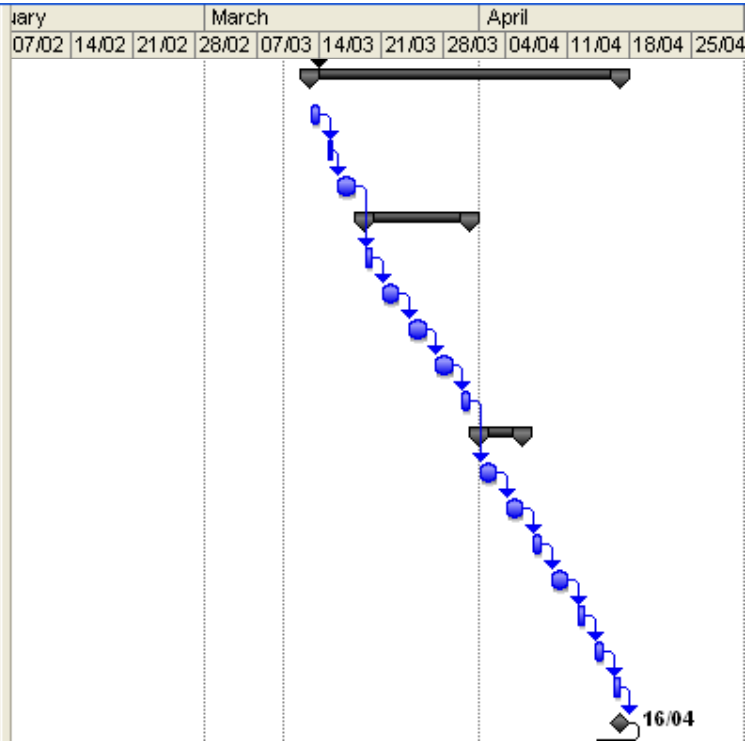


Figura VI: Plan de proyecto, PEC2.

Fase 3, desarrollo y pruebas (PEC3):

Task Name	Duration	Start	Finish
03 - Desarrollo y pruebas (PEC3)	**35 days**	**Sun 17/04/11**	**Sat 21/05/11**
03.01 - Desarrollo generador: Tratamiento fichero de	8 days	Sun 17/04/11	Sun 24/04/11
03.02 - Desarrollo generador: Contenido semántico	5 days	Mon 25/04/11	Fri 29/04/11
03.03 - Desarrollo generador: Figuras canvas	8 days	Sat 30/04/11	Sat 07/05/11
03.04 - Desarrollo generador: Otras características y	5 days	Sun 08/05/11	Thu 12/05/11
03.05 - Pruebas integradas	8 days	Fri 13/05/11	Fri 20/05/11
03.06 - Entrega desarrollo y pruebas (PEC3)	1 day	Sat 21/05/11	Sat 21/05/11
Fin desarrollo y pruebas (PEC3)	0 days	Sat 21/05/11	Sat 21/05/11

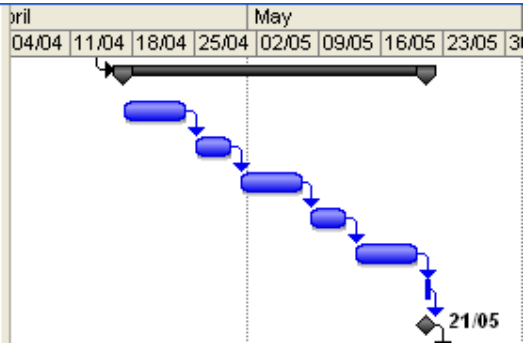


Figura VII: Plan de proyecto, PEC3.

Fase 4, elaboración de la memoria y presentación final:

Task Name	Duration	Start	Finish
04 - Elaboración memoria y presentación del PFC (E	**26 days**	**Sun 22/05/11**	**Thu 16/06/11**
04.01 - Elaboración memoria del PFC	10 days	Sun 22/05/11	Tue 31/05/11
04.02 - Elaboración de la presentación del PFC	15 days	Wed 01/06/11	Wed 15/06/11
04.03 - Entrega final	1 day	Thu 16/06/11	Thu 16/06/11
Fin elaboración memoria y presentación final	0 days	Thu 16/06/11	Thu 16/06/11
Fin PFC	0 days	Thu 16/06/11	Thu 16/06/11

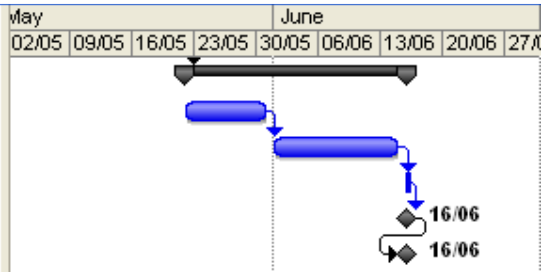


Figura VIII: Plan de proyecto, memoria y presentación final.

Por último, presentamos el diagrama gantt completo:

Task Name	Duration	Start	Finish
Inicio PFC	0 days	Wed 02/03/11	Wed 02/03/11
01 - Plan de trabajo (PEC1)	**11 days**	**Wed 02/03/11**	**Sat 12/03/11**
01.01 - Selección enfoque PFC	4 days	Wed 02/03/11	Sat 05/03/11
01.02 - Elaboración plan de trabajo PFC	6 days	Sun 06/03/11	Fri 11/03/11
01.03 - Entrega plan de trabajo PFC	1 day	Sat 12/03/11	Sat 12/03/11
Fin Plan de trabajo (PEC1)	0 days	Sat 12/03/11	Sat 12/03/11
02 - Análisis y diseño (PEC2)	**35 days**	**Sun 13/03/11**	**Sat 16/04/11**
02.01 - Análisis formato PGN	2 days	Sun 13/03/11	Mon 14/03/11
02.02 - Análisis formato FEN	1 day	Tue 15/03/11	Tue 15/03/11
02.03 - Análisis y diseño de uso de librerías LEX/CUF	3 days	Wed 16/03/11	Fri 18/03/11
02.04 - Análisis lenguaje OWL	**13 days**	**Sat 19/03/11**	**Thu 31/03/11**
02.04.01 - Introducción OWL	2 days	Sat 19/03/11	Sun 20/03/11
02.04.02 - OWL y RDF/RDFS	3 days	Mon 21/03/11	Wed 23/03/11
02.04.03 - Sublenguajes OWL	3 days	Thu 24/03/11	Sat 26/03/11
02.04.04 - Descripción del lenguaje OWL	3 days	Sun 27/03/11	Tue 29/03/11
02.04.05 - Las capas de OWL	2 days	Wed 30/03/11	Thu 31/03/11
02.05 - Diseño ontología de ajedrez con Protég	**6 days**	**Fri 01/04/11**	**Wed 06/04/11**
02.05.01 - Descripción herramienta Protégé	3 days	Fri 01/04/11	Sun 03/04/11
02.05.02 - Diseño ontología de ajedrez	3 days	Mon 04/04/11	Wed 06/04/11
02.06 - Análisis y diseño de etiquetas semánticas HT	2 days	Thu 07/04/11	Fri 08/04/11
02.07 - Análisis y diseño canvas en HTML5	3 days	Sat 09/04/11	Mon 11/04/11
02.08 - Otras características HTML requeridas	2 days	Tue 12/04/11	Wed 13/04/11
02.09 - Diseño del generador semántico HTML5	2 days	Thu 14/04/11	Fri 15/04/11
02.10 - Entrega análisis y diseño (PEC2)	1 day	Sat 16/04/11	Sat 16/04/11
Fin análisis y diseño (PEC2)	0 days	Sat 16/04/11	Sat 16/04/11
03 - Desarrollo y pruebas (PEC3)	**35 days**	**Sun 17/04/11**	**Sat 21/05/11**
03.01 - Desarrollo generador: Tratamiento fichero de	8 days	Sun 17/04/11	Sun 24/04/11
03.02 - Desarrollo generador: Contenido semántico	5 days	Mon 25/04/11	Fri 29/04/11
03.03 - Desarrollo generador: Figuras canvas	8 days	Sat 30/04/11	Sat 07/05/11
03.04 - Desarrollo generador: Otras características y	5 days	Sun 08/05/11	Thu 12/05/11
03.05 - Pruebas integradas	8 days	Fri 13/05/11	Fri 20/05/11
03.06 - Entrega desarrollo y pruebas (PEC3)	1 day	Sat 21/05/11	Sat 21/05/11
Fin desarrollo y pruebas (PEC3)	0 days	Sat 21/05/11	Sat 21/05/11
04 - Elaboración memoria y presentación del PFC (E	**26 days**	**Sun 22/05/11**	**Thu 16/06/11**
04.01 - Elaboración memoria del PFC	10 days	Sun 22/05/11	Tue 31/05/11
04.02 - Elaboración de la presentación del PFC	15 days	Wed 01/06/11	Wed 15/06/11
04.03 - Entrega final	1 day	Thu 16/06/11	Thu 16/06/11
Fin elaboración memoria y presentación final	0 days	Thu 16/06/11	Thu 16/06/11
Fin PFC	0 days	Thu 16/06/11	Thu 16/06/11

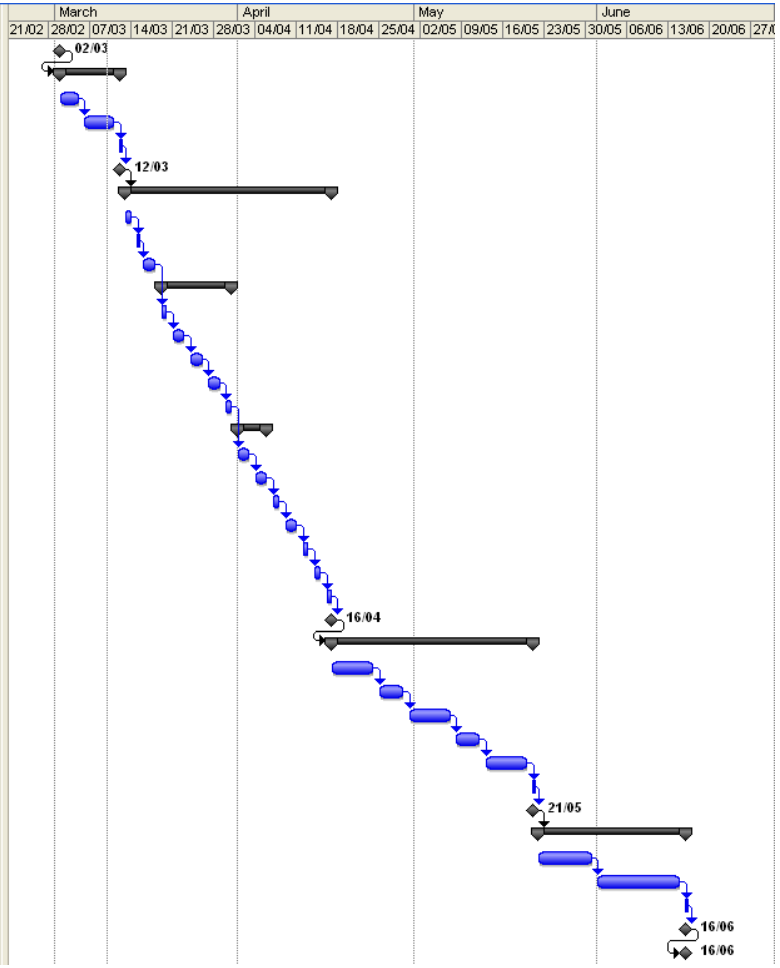


Figura IX: Plan de proyecto detallado.

En base a esta planificación, el PFC terminaría el 16 de junio de 2011, fecha inicialmente prevista por la UOC para su finalización.

0.8 Hoja de resumen del proyecto

En esta última sección se incluye el cuadro resumen del proyecto con los hitos principales del mismo:

Resumen	
Hito	Fecha
Entrega Plan de Proyecto	12 de marzo de 2011
Entrega Análisis y diseño	16 de abril de 2011
Entrega desarrollo y pruebas	21 de mayo de 2011
Entrega memoria final y presentación	16 de junio de 2011

Tabla II: cuadro resumen hitos del proyecto

1 OBJETIVOS Y CONCEPTOS GENERALES

En este primer capítulo de la memoria del Proyecto Final de Carrera (PFC en adelante) se presentarán los objetivos principales definidos para el proyecto y se analizarán los conceptos generales que serán el punto de partida a la hora de desarrollar el siguiente capítulo del proyecto, el análisis, diseño e implementación de la solución. Los apartados que han sido definidos son los siguientes:

Contenido	Apartado	Comentarios
Objetivos del PFC	1.1	Descripción de los objetivos principales del proyecto final de carrera.
Conceptos generales	1.2	Descripción de los conceptos generales del proyecto, información necesaria para poder afrontar el desarrollo de los siguientes capítulos de la memoria.

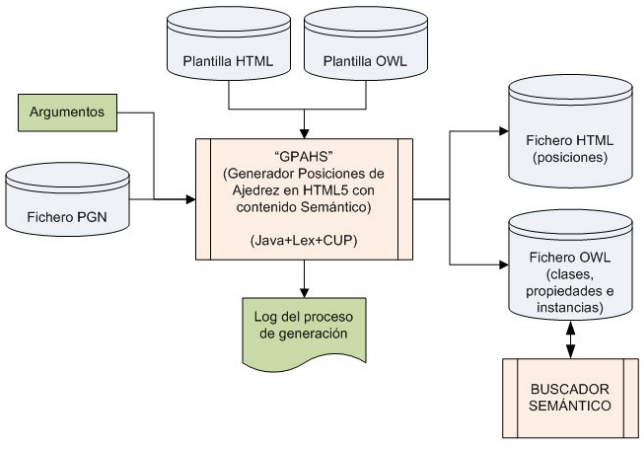
Tabla III: apartados del capítulo 1.

1.1 Objetivos del PFC

El objetivo principal del PFC es el siguiente:

Análisis, diseño y desarrollo de un aplicativo que a partir de un fichero con formato "PGN" genere dos tipos distintos de fichero, el primero será una página HTML con contenido gráfico que represente las posiciones, soluciones y datos complementarios de cada posición, y el segundo fichero contendrá una base de conocimiento en formato OWL, ontología compuesta de clases y propiedades, previamente definidas con Protégé, y de individuos que serán creados dinámicamente a partir de la información contenida en el fichero PGN de entrada.

A este aplicativo lo hemos denominado "*Generador de Posiciones de Ajedrez en HTML y contenido Semántico*", y hemos utilizado el acrónimo **GPAHS** a lo largo de la memoria. A continuación, comentaremos los detalles de este objetivo.



A modo de resumen inicial, En la figura adjunta podemos ver las entradas y salidas previstas en nuestro proceso. Las entradas son las siguientes:

- Argumentos de ejecución del proceso, los cuales permiten personalizar cada ejecución.

- Fichero PGN, con las posiciones didácticas de ajedrez (posición de piezas y solución con el mejor movimiento).

- Plantilla HTML que contendrá toda la información estática de las páginas HTML a generar.

Figura X: Generador GPAHS, esquema general.

- Plantilla OWL que contendrá las clases y propiedades que serán definidas con el editor Protégé.

Las salidas del proceso son las siguientes:

- Log del proceso de ejecución. Sin activación de traza (uno de los argumentos) debe presentar 3 mensajes:

  - Tipo de finalización del proceso (satisfactoria o no),

  - Número de posiciones generadas en la página HTML, y

  - Número de instancias creadas en la base de conocimiento.

En el caso de activar la traza de ejecución, se debe presentar toda la información significativa del fichero PGN de entrada (la contenida en la memoria intermedia del proceso).

- Fichero HTML generado con las posiciones didácticas propuestas en el fichero PGN de entrada.

- Fichero OWL generado con las clases, propiedades (datos estáticos) e instancias (datos dinámicos). La base de conocimiento generada podría ser tratada por un buscador semántico para realizar búsquedas de las posiciones didácticas que se ajusten a las necesidades de cualquier usuario en cada momento.

Parece lógico pensar que nuestro aplicativo a desarrollar guarda bastantes semejanzas con los compiladores, traductores e intérpretes analizados en las asignaturas de compiladores I y II, dado que tenemos que leer un fichero en un formato determinado (fichero PGN) y transformarlo en dos ficheros, el primero con formato “HTML5” y el segundo con formato “OWL”. Por ello, en la primera sección del siguiente apartado revisaremos los componentes básicos de un programa de estas características, dado que esta arquitectura de programa será la que utilizaremos en nuestro generador.

La tarea de crear un traductor de un lenguaje fuente a uno o varios lenguajes objeto es compleja, pero se puede reducir en gran medida dicha complejidad si el proceso se divide en fases especializadas que realicen cada una de estas tareas específicas. La estructura de un proceso traductor de código (también válida para otro tipo de procesos: compiladores, etc.) es la siguiente:

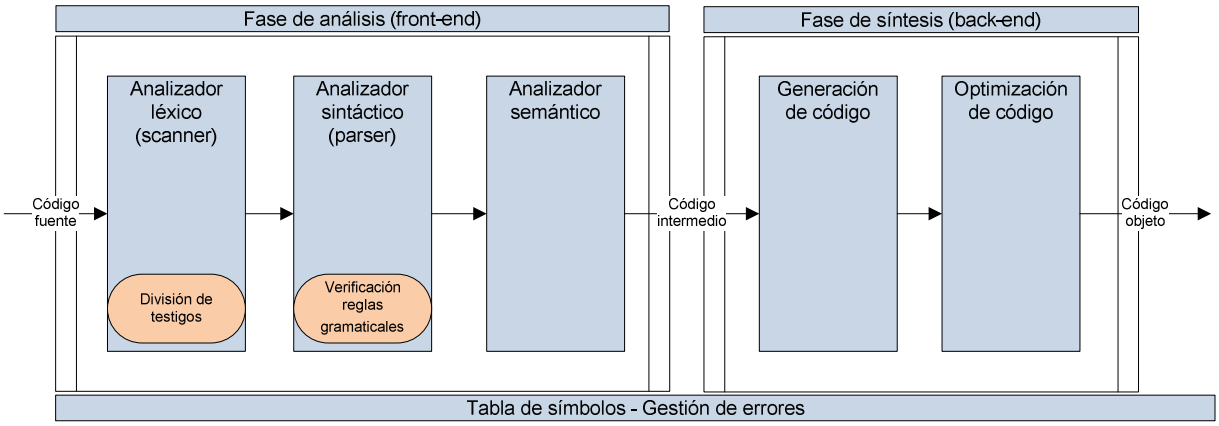


Figura XI: Fases de un compilador.

Como se puede ver en la figura anterior, las actividades de un traductor se dividen en dos fases genéricas:

- Fase de análisis (front-end): Analiza el programa fuente (análisis léxico, sintáctico y semántico), se buscan errores en el origen de la información, y se genera un código intermedio que será la entrada de la siguiente fase. Estas sub-fases dependen del lenguaje fuente y deben ser independientes del código objeto a generar, y de la máquina donde será ejecutado el traductor, excepto en la primera parte del proceso donde se lee el código fuente (analizador léxico).
- Fase de síntesis (back-end): A partir del código intermedio generado por la fase anterior, se genera y optimiza (si procede) el código objeto resultante. Estas actividades dependen del código objeto a generar y en ocasiones (aunque no es nuestro caso) de la máquina donde se ejecutará el código generado.

La primera fase, durante el análisis léxico-sintáctico-semántico del fichero fuente, se generará información en memoria sobre una estructura que se denomina *tabla de símbolos*. Igualmente, durante todo el proceso de traducción se realiza una gestión de errores, que podrán ser mostrados durante la ejecución del proceso.

A modo de resumen final, mostramos en la siguiente figura cada uno de los elementos principales del aplicativo a desarrollar, con una breve descripción incluida a continuación.

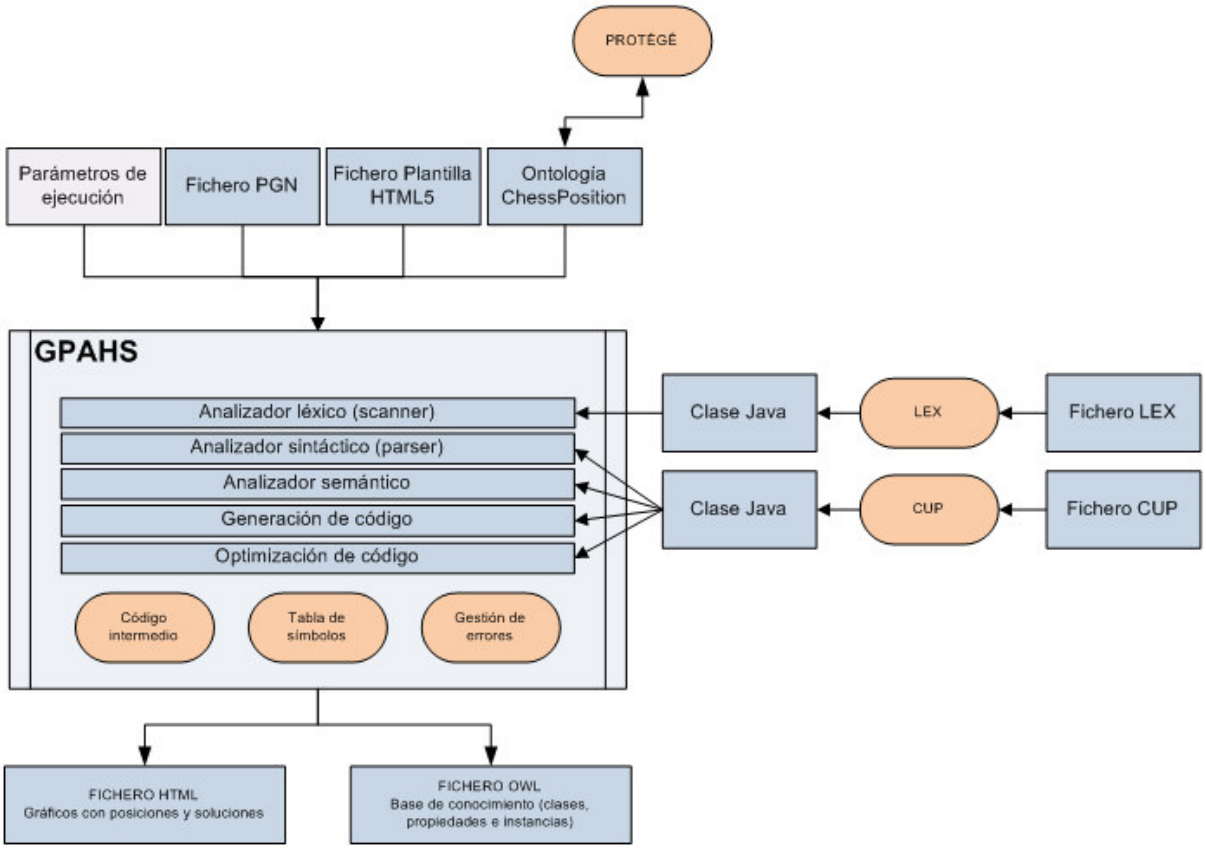


Figura XII: Elementos del generador GPAHS.

Elementos principales del generador GPAHS:

- Fichero PGN: Fichero de entrada del generador, con formato PGN. Este fichero contiene las posiciones de ajedrez didácticas que deseamos transformar en contenido HTML y en contenido semántico (base de conocimiento).
- Fichero LEX: Fichero fuente con la definición léxica inicial, elemento principal del analizador léxico del generador.
- Fichero CUP: Fichero fuente con la definición sintáctica inicial, elemento principal del analizador sintáctico del generador. Además, sobre este fichero se incluirán todos atributos y clases necesarias para el resto de actividades del generador.
- Herramienta LEX: Utilidad que convierte ficheros "JLex" en clases java que serán utilizadas por el generador.
- Herramienta CUP: Utilidad que convierte ficheros "CUP" en clases java. El programa principal del generador se corresponde con el fichero que obtenemos en esta fase de compilación.
- Herramienta Protégé: Editor de ontologías OWL. Permitirá definir nuestra ontología (clases, propiedades y facetas). También será utilizada en la fase de pruebas para validar la correcta creación de individuos para dicha ontología.
- Ontología ChessPosition: Ontología creada con Protégé, la cual representa el modelo que hemos conceptualizado. Los individuos a generar deberán pertenecer a este dominio.
- Fichero Plantilla HTML5: Plantilla HTML con la información estática, que nos servirá como punto de partida a la hora de desarrollar la fase de generación de código HTML de nuestro generador.
- Fichero HTML: Fichero de salida de nuestro generador. Contendrá una figura gráfica por cada posición del fichero de entrada, junto a su solución e información complementaria.
- Fichero OWL: Fichero de salida de nuestro generador. Contendrá lo que denominamos "base de conocimiento", compuesto por las clases y propiedades de la ontología de referencia, y los individuos (uno por cada posición) generados a partir de la información del fichero de entrada.
- Parámetros de ejecución: El generador deberá aceptar parámetros para personalizar el modo de ejecución: nombre del fichero PGN de entrada, nombre del fichero plantilla HTML, nombre de la ontología de referencia, nombre de la clase asociada a las posiciones contenidas en el fichero PGN y, por último, tipo de traza.
- Generador GPAHS: Como ya hemos comentado, programa Java desarrollado utilizando las fases de ejecución tradicionales de un traductor o compilador, fase de análisis con el analizador léxico, sintáctico y semántico; y fase de síntesis con el generador y optimizador de código.
- Herramientas "java" y "javac" (no incluidas en la figura): Por último, requerimos el uso de las herramientas "javac" para obtener el *byte-code*, y "java" para poder ejecutar algunos de los elementos (programas) mostrados en la figura anterior.

1.2 Conceptos generales

En este apartado se realizará una presentación de los conceptos generales relacionados con el objetivo del PFC, cuya comprensión es necesaria para el desarrollo de la fase de análisis y diseño de los elementos que conformarán la implementación del generador **GPAHS**. Las secciones contenidas en este apartado son los siguientes:

Contenido	Sección	Comentarios
Arquitectura del generador	1.2.1	Como ya hemos comentado, nuestro generador tendrá la estructura tradicional de un traductor. En esta sección se analizarán las fases de análisis y síntesis de este tipo de procesos, y de los componentes que conforman cada fase.
Formato de representación de partidas PGN	1.2.2	Para poder interpretar un fichero en formato "PGN" debemos comprender su contenido. En esta sección se analizarán las distintas partes que conforman un fichero PGN, las cuales serán tratadas en la fase de análisis por cada uno de los elementos del generador (analizador léxico, analizador sintáctico, y analizador semántico).
Formato de representación de posiciones FEN	1.2.3	Uno de los campos más importantes de un fichero PGN, dentro de los objetivos de este PFC, es el que indica una posición determinada de una partida de ajedrez. El formato asociado con esta información es el denominado "notación FEN". En esta sección se analizarán en detalle las diferentes partes contenidas en este formato.
Ontología y lenguaje OWL	1.2.4	La parte más importantes de este PFC, desde el punto de vista didáctico, es la generación automática de contenido semántico. En esta sección se analizarán las distintas características del lenguaje OWL, desde una introducción del mismo y definición de los sub-lenguajes existentes, hasta el detalle de cada elemento sintáctico del sub-lenguaje OWL Lite, sub-lenguaje utilizado tanto en la definición de nuestra ontología como en la creación automática de instancias o individuos.
Lenguaje HTML5	1.2.5	En esta sección última sección analizaremos las partes del lenguaje HTML, en su versión 5, que nuestro programa deberá generar o utilizar, pero sólo desde la perspectiva de nuestras necesidades de desarrollo. Por ello, veremos una introducción de los elementos semánticos que nos permiten estructurar el contenido HTML, los elementos que nos permiten crear dinamismo en la página y, por último, los elementos que nos permiten añadir contenido gráfico en la página.

Tabla IV: Secciones del apartado 1.2.

1.2.1 Arquitectura del generador

Los puntos tratados en esta sección son los siguientes:

- Analizador léxico.
- Analizador sintáctico.
- Analizador semántico.
- Generación de código intermedio.
- Optimización de código.
- Generación de código.

Analizador léxico

Un analizador léxico tiene como objetivo principal la lectura de un fichero fuente, carácter a carácter, para formar grupos de caracteres, denominados "tokens", "lexemas" o testigos, con un significado léxico mínimo (se mantendrá en este documento el término en inglés por ser más habitual en el ámbito de compiladores). Normalmente, estos tokens son tratados como una entidad única. El analizador léxico también elimina los componentes no esenciales del programa fuente ignorando, por ejemplo, los espacios en blanco, tabuladores, caracteres de final de línea, comentarios y, en general, todo lo que no sea necesario para el resto de actividades del traductor.

A modo de ejemplo, y utilizando información contenida en todo fichero PGN, analicemos algunos testigos que un analizador léxico debería detectar:

[Date "2011.03.13"]

Token	Valor
CORCHETE_ABRIR	[
CADENASTRING	"2011.03.13"
CORCHETE_CERRAR	]

NOTA: *Sería posible detectar más tokens individuales, como por ejemplo la "doble comilla" (") pero como veremos en los siguientes apartados, es más interesante en nuestro proyecto tratar todo contenido encerrado entre comillas dobles como un único string*.

Para poder diseñar correctamente un analizador léxico en nuestro proyecto, necesitamos conocer la composición completa de los ficheros PGN. Analicemos, por lo tanto, su contenido.

Los ficheros PGN están compuestos de datos en modo carácter (no son ficheros binarios) formando tokens. Un token es una secuencia contigua de caracteres que representa una unidad léxica básica. Los tokens están separados entre sí por un número indeterminado (variable) de "espacios en blanco". Por espacio en blanco entendemos en esta memoria tanto los caracteres de espacio en blanco en sí, como los tabuladores o los caracteres de salto de línea.

En el formato PGN, los datos están organizados como líneas de texto simple sin ningún tipo de caracteres especiales, o marcadores de estructura de los registros que hayan sido impuestas por los sistemas operativos subyacentes. En el formato "import" (este formato se analiza en el siguiente apartado), las líneas de texto tienen un límite de 255 caracteres, incluyendo el carácter especial de salto de línea. No obstante, se recomienda no utilizar líneas de más de 80 caracteres para que todo tipo de editor pueda tratar estos ficheros. En algunas ocasiones, algunas etiquetas pueden tener un contenido muy largo, como la etiqueta FEN (igualmente, se analizará en un apartado posterior) pero en raras ocasiones se sobrepasa.

Un fichero PGN está compuesto de una colección de cero o más partidas en formato PGN. Un fichero PGN, por lo tanto, podría estar vacío de contenido, aunque sería una base de datos sin información. Una partida PGN, a su vez, está compuesta de dos secciones. La primera sección contiene parejas de "etiqueta-valor", y la segunda sección contiene los movimientos de la partida o solución de una posición. Un par "etiqueta-valor" proporciona información que identifica la partida gracias a parámetros estándar. La sección de movimientos suele contener los movimientos realizados de forma numerada, así como posibles anotaciones de movimientos, concluyendo también en la mayoría de las ocasiones con el resultado de la partida. Para representar los movimientos de una partida de ajedrez se utiliza la *notación SAN* (Standard Algebraic Notation), normalmente referenciada en castellano como "notación algebraica".

La sección "etiqueta-valor" está compuesta de una serie de cero o más pares de etiquetas-valor. Un par etiqueta-valor está compuesto de cuatro tokens consecutivos:

- Un token "corchete" del lado izquierdo.
- Un token simbólico.
- Un token de tipo string.
- Un token "corchete" del lado derecho.

El token simbólico es el nombre de la etiqueta y el token de tipo string contiene el valor asociado con la etiqueta anterior. A continuación se verán todas las etiquetas válidas dentro de un fichero PGN. En este punto sólo resaltaremos que una etiqueta sólo puede aparecer una vez dentro de una misma partida.

Además, los nombres de etiquetas sólo pueden estar compuestos de letras, dígitos numéricos, y el carácter "guión bajo" (_). Esta restricción se definió para facilitar el mapeo entre los nombres de las etiquetas y los nombres de atributos de programas de bases de datos de propósito general.

En los ficheros con formato "import", podría haber cero o más de un espacio en blanco entre los token que conforman el par "etiqueta-valor". En cambio, en el formato "export", no puede haber espacios en blanco entre el corchete izquierdo ([) y el nombre de etiqueta; y entre el campo valor y el corchete de la derecha (]); permitiéndose un único espacio en blanco entre el nombre de la etiqueta y el campo valor.

Los nombres de etiquetas, así como el resto de símbolos, son "case sensitive", es decir, son casos distintos un carácter escrito en mayúsculas y minúsculas.

En el formato "import", un fichero puede contener múltiples parejas "etiqueta-valor" sobre la misma línea, incluso un par "etiqueta-valor" podría ocupar más de una línea. En cambio, en el formato "export", cada pareja "etiqueta-valor" debe ocupar una única línea, justificando todo su contenido a

la izquierda, incluyendo una línea en blanco completa tras la última pareja “etiqueta-valor”, anterior a la entrada de los movimientos de la partida.

Algunos valores de etiquetas pueden estar compuestos de una secuencia de elementos. Por ejemplo, los jugadores de una partida podrían ser más de uno por cada bando (por ejemplo, en partidas de exhibición). Cuando esto ocurre, se incluye un carácter “dos puntos” (:) entre elementos los adyacentes. Obviamente, dado que el carácter “dos puntos” (:) tiene un significado especial, no se puede incluir dicho carácter como parte de un string.

En las primeras definiciones de formato PGN, el campo valor siempre debía corresponderse con un string. Así sucede, como veremos, en los campos STR (Seven Tag Roster), y esta circunstancia no cambiará. Sin embargo, el formato PGN actual acepta poder ser expandido y ya existen etiquetas no-STR que pueden aceptar otro tipo de valores, respetando siempre la secuencia vista anteriormente: “corchete izquierdo” ([), etiqueta, valor, y “corchete derecho” (]).

A continuación veremos cada uno de los posibles tipos de tokens.

Tokens tipo string

Los tokens de tipo string son una secuencia de cero o más caracteres imprimibles delimitados por un par de caracteres de “comilla doble” (valor ASCII: 34, valor hexadecimal: 0x22). Las características de este tipo de tokens son las siguientes:

- Un string vacío es válido, y estaría representado por dos dobles comillas adyacentes. En este punto hay que resaltar que un apóstrofe, aunque sea repetido y sea cual sea su tipo (``, ´´, etc.) no es lo mismo que una doble comilla (“) y, por ello, no sería válido su uso para representar un string dentro de un fichero PGN.
- Para incluir una doble comilla (“) dentro de un string debemos anteponer una barra invertida (\) a la doble comilla (“), sin espacios en blanco entre ambos.
- A su vez, Para poder incluir una barra invertida (\) dentro de un string deberemos incluir una nueva barra invertida (\) por delante de la primera, sin espacios en blanco entre ambas.
- Los strings se utilizan para asociar valores a cada una de las etiquetas válidas en un fichero PGN.
- Los caracteres no imprimibles, como los tabuladores o los saltos de línea no son válidos dentro de un string.
- Como ya se ha indicado, un string termina con una doble comilla (“).
- En la versión actual, los strings están limitados a un máximo de 255 caracteres.

Tokens tipo entero

Un token de tipo entero es una secuencia de uno o más caracteres de dígitos decimales. Los tokens de tipo entero se utilizan como ayuda de representación, para indicar el número de movimiento dentro de una partida. Este tipo de token termina cuando se alcanza un carácter que no representa un dígito decimal.

Token especial “punto” (.)

El carácter “punto” (.) es en sí mismo un token, y se utiliza como delimitador de los números de movimiento, dentro de una partida. Es, por ello, un token delimitador.

Token especial “asterisco” (*)

El carácter “asterisco” (*) es en sí mismo un token, y se utiliza como una de las posibles marcas de finalización de una partida. Se utiliza para indicar una partida incompleta, o también para indicar una partida con resultado desconocido o no significativo. Es, por ello, un token delimitador.

Tokens especiales “corchete” ([o])

Los caracteres “corchete” ([o]) son en sí mismos tokens. Se utilizan para delimitar los pares “etiqueta-valor” que hemos visto anteriormente. Son, por ello, tokens delimitadores.

Tokens especiales “paréntesis” ((o))

Los caracteres “paréntesis” ((o)) son en sí mismos tokens. Se utilizan para delimitar variantes de juego, dentro de una partida. Son, por ello, tokens delimitadores.

Tokens especiales “menor” y “mayor” (< o >)

Los caracteres “menor” y mayor” (< o >) son en sí mismos tokens. En la actualidad no se utilizan, pero están reservados para un uso futuro. Serán también tokens delimitadores, cuando proceda, de algún otro elemento.

Token NAG

El NAG (Numeric Annotation Glyph – Glifos de anotación numérica) es un token compuesto del carácter “dólar” ($) seguido (sin espacios en blanco) por uno o más caracteres numéricos. El token termina cuando aparece el primer (cualquier) carácter no numérico.

NOTA: *Se ha incluido un anexo en esta memoria con todos los tokens válidos actualmente.*

Token simbólico

Un token simbólico comienza por un carácter de tipo letra o numérico, seguido por una secuencia de cero o más caracteres de tipo simbólico (letras y números). Los caracteres de tipo letra son de la “a” a la “z”, y de la “A” a la “Z”; y los dígitos numéricos son del “0” al “9”. También son válidos como carácter de continuación (todos menos el primero) el “guión bajo” (_), el símbolo “más” (+), el símbolo “almohadilla” (#), el símbolo “igual” (=), el símbolo “dos puntos” (:), y el símbolo “guión” (-).

Todos los caracteres de un token simbólico son significativos, terminando con el primer carácter no simbólico que aparezca. Estos tokens se utilizan para varios propósitos y, en la actualidad, existe un límite de 255 caracteres en su longitud.

Analizador sintáctico

El analizador sintáctico utiliza los testigos proporcionados por el analizador léxico y comprueba si llegan en el orden correcto. Para ello, se debe definir la gramática de contexto que define el lenguaje fuente. La salida del analizador sintáctico suele ser un árbol con la estructura sintáctica del programa fuente.

Por ejemplo, para la siguiente entrada de un fichero PGN:

[Event "Polgar, Z - 200 Mattkombinationen"]

Se podría generar el siguiente árbol sintáctico:

atributo → CORCHETE_ABRIR característica CORCHETE_CERRAR

característica → car_event

car_event → ETIQUETA_EVENT STRING

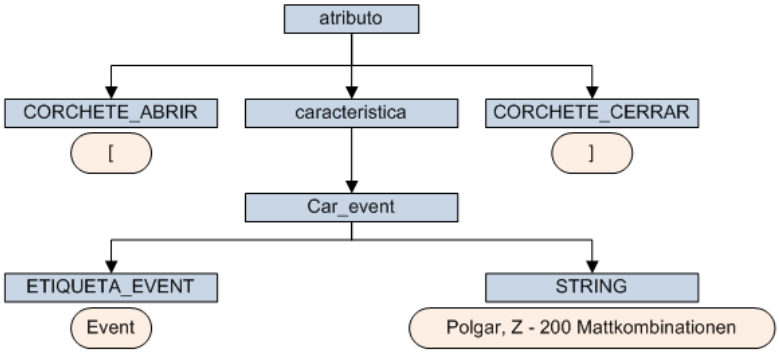


Figura XIII: Árbol sintáctico.

Analizador semántico

El analizador semántico se ocupa de comprobar el significado de las sentencias. Puede haber sentencias sintácticamente correctas, pero que no se puedan ejecutar por no tener ningún sentido. Generalmente, este análisis se hace al mismo tiempo que el sintáctico, e introduce unas rutinas semánticas que intentan encontrar errores de significado (semánticos) a partir del árbol sintáctico, y al mismo tiempo reúnen información sobre los tipos de datos de los objetos del programa fuente (variables, constantes, etc.) que serán utilizados en la fase de generación de código.

En el siguiente apartado se presentará el contenido y formato completo de los ficheros PGN. A modo de ejemplo, incluimos a continuación una posible validación a realizar.

La etiqueta “Date” contiene la fecha en la que se ha desarrollado la partida contenida en el fichero PGN. El string asociado a esta etiqueta contiene una fecha con el formato “AAAA.MM.DD”, siendo “AAAA” el año, “MM” el mes, y “DD” el día. Tanto el año, el mes como el día son dígitos numéricos.

Pero también se acepta el carácter especial "?" cuando desconocemos parte de la fecha pero en el orden apropiado. Con algunos ejemplos se verá más clara la validación semántica a realizar:

- [Date "2011.04.03"] – Fecha completa conocida.
- [Date "2011.04.??"] – Desconocemos el día, pero sí el año y mes.
- [Date "2011. ??.??"] – Desconocemos el día y mes pero conocemos el año.
- [Date "????.??.??"] – Desconocemos la fecha de desarrollo de la partida.

Los ejemplos anteriores representan todas las posibilidades válidas de representación de fechas. Por ello, cualquier otra combinación de fechas sería incorrecta:

- [Date "****.**.**"] – Formato Incorrecto, el "*" no es válido.
- [Date "????.13.13"] – Formato Incorrecto, si desconocemos el año, el resto de información (mes y día) no es procedente.
- [Date "2011.??.31"] – Formato Incorrecto, si desconocemos el mes, indicar el día es incorrecto.

En el siguiente apartado veremos el formato completo de los ficheros PGN, donde se incluyen todas las validaciones semánticas a realizar en nuestro generador.

Generación de código intermedio

La generación del código intermedio transforma el árbol sintáctico/semántico en una representación equivalente en un lenguaje intermedio independiente de la máquina para la cual se genera el código objeto, y también independiente del lenguaje objeto a generar. Este código intermedio suele disponer de un juego de instrucciones lo bastante simple como para poder generar posteriormente el código objeto de manera sencilla.

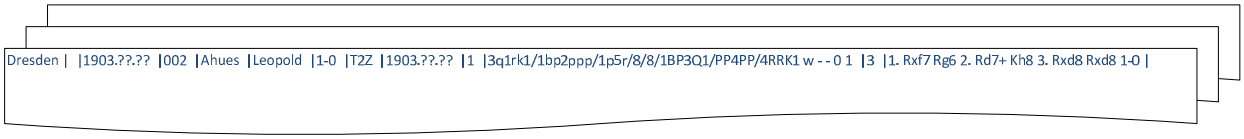
Nuestro generador almacenará la información de todas las partida contenidas en el fichero PGN en tablas Java (por ejemplo, "hashtables") donde cada registro representará toda la información asociada a cada partida.

A modo de ejemplo y de una manera gráfica, veamos cómo se podría almacenar la información de la siguiente partida:

```
[Event "Dresden"]
[Site ""]
[Date "1903.??.??"]
[Round "002"]
[White "Ahues"]
[Black "Leopold"]
[Result "1-0"]
[Annotator "T2Z"]
[EventDate "1903.??.??"]
[SetUp "1"]
[FEN "3q1rk1/1bp2ppp/1p5r/8/8/1BP3Q1/PP4PP/4RRK1 w - - 0 1"]
[PlyCount "3"]

1. Rxf7 Rg6 2. Rd7+ Kh8 3. Rxd8 Rxd8 1-0
```

Tabla HASHTABLE



Cada registro contiene toda la información relevante de una partida, sin ningún tipo de conversión o manipulación de la información.

Optimización de código

En la fase de síntesis, cuando procede, se realiza la optimización del código, con el objetivo de generar un código objeto más eficiente. Por ejemplo, la optimización podría ir dirigida a:

- Reducción del espacio ocupado por el código generado.
- Mejora del tiempo de ejecución.
- Menor uso de memoria.

Obviamente, estas mejoras son, en la mayoría de las ocasiones, incompatibles entre sí, dado que en la mayoría de las ocasiones se consigue, por ejemplo, una mejora del tiempo de ejecución haciendo un uso mayor de la memoria, y viceversa.

Las optimizaciones de código se pueden realizar sobre dos tipos de código:

- Sobre el código intermedio, generado al final de la fase de análisis. Estas mejoras son independientes de la máquina de ejecución.
- Sobre el código objeto, con mejoras dependientes (o independientes) de la máquina de ejecución.

Nuestro generador es simple, en el sentido de que el código generado tiene siempre el mismo formato (código HTML5) y será desarrollado, en una primera instancia, de forma eficiente. No obstante, se incluirá un proceso relacionado con esta fase consistente en la eliminación de partidas duplicadas, en el caso de que existan, a partir de la posición inicial de la posición concreta de la partida.

Generación de código

En esta última fase del proceso de compilación, se genera el código objeto. En nuestro caso, no se tratará de código realmente “objeto”, en el sentido que no generaremos código máquina que podrá ser ejecutado sobre un sistema operativo concreto.

Nuestro aplicativo generará código HTML5 que podrá ser visualizado con cualquier navegador que acepte las nuevas características del lenguaje. Como veremos en uno de los últimos apartados de este capítulo, el código HTML5 a generar contendrá etiquetas que nos permitan:

- Estructurar el contenido de la página.
- Crear dinamismo.
- Crear elementos gráficos.

Contendrá elementos (individuos) OWL.

1.2.2 Formato de representación de partidas PGN

La "Notación Portátil de Juego" (en inglés: Portable Game Notation – "PGN" en adelante) es un formato de ficheros ASCII de ordenador que permite grabar partidas de ajedrez, tanto los movimientos realizados en la partida, como toda la información relacionada: jugadores, fecha, torneo, ELO de los jugadores, etc. La mayoría de los programas de ordenador de ajedrez reconocen este formato, debido a que es muy sencillo su manejo y manipulación automática o manual.

El formato "PGN" (la extensión del fichero con este formato es ".pgn") está estructurado de forma que su lectura y escritura es sencilla directamente por usuarios humanos y, en consecuencia, por programas informáticos. Las jugadas de la partida están anotadas en una notación denominada "algebraica", formato que veremos a continuación.

Existen dos sub-formatos dentro de la especificación PGN, el formato de "importación" (import) y el formato de "exportación" (export). El primer formato describe información que ha sido preparada (habitualmente) a mano y es, por ello, intencionadamente más flexible que el segundo formato. Un programa preparado para leer datos en formato PGN también debe estar preparado para leer esta variante. El formato de "exportación" (export) es más estricto, está compuesto de información que normalmente ha sido generada bajo el control de un programa informático. Podría decirse que se trata del "código fuente" de una partida de ajedrez, con sus reglas y restricciones completas. Los ficheros generados por cualquier programa que los sepa utilizar correctamente deberían ser iguales entre sí, byte a byte.

Un fichero PGN está compuesto de un número variable de partidas de ajedrez. Cada partida individual contiene dos partes:

- La primera parte contiene un conjunto de pares de datos, compuesto cada par del nombre de una etiqueta y el valor asociado a dicha etiqueta. Esta parte del fichero contiene toda la información relevante.
- La segunda parte son la secuencia de las jugadas, que representan los movimientos de una partida de ajedrez, con la posibilidad de incluir comentarios opcionales.

Sintácticamente hablando, un fichero PGN podría no contener partidas, pero sería un fichero sin información, por lo tanto, sin utilidad.

Sección 1 fichero PGN: parejas "etiquetas-valor"

Toda la información asociada a una partida está representada como parejas de valores con el siguiente formato:

- "[": carácter obligatorio. Representa el inicio de un dato de la partida.
- Nombre de la etiqueta: A continuación veremos todas las etiquetas aceptadas por el formato PGN.
- String: valor asociado a una etiqueta, siempre encerrado entre comillas dobles (").
- "]": carácter obligatorio. Representa el fin de un dato de la partida.

Para que un fichero pueda decirse que cumple con el formato PGN debe ser almacenado obligatoriamente con las etiquetas "STR" ("Seven Tag Roster" – "lista de las siete etiquetas"), aunque desconozcamos el valor real asociado a cada etiqueta. Como veremos a continuación, existen caracteres "comodín" que podemos utilizar como string asociado a estas etiquetas cuando desconocemos el valor. Para el resto de etiquetas no obligatorias, si desconocemos un valor, simplemente, no se incluye la etiqueta.

En el formato import el orden de estas etiquetas no es importante, pero en el formato export, estas siete etiquetas deben aparecer antes que cualquier otra etiqueta, en el mismo orden que presentamos a continuación:

Etiqueta	Significado
Event	Evento. Nombre del torneo o de la competición donde se ha celebrado la partida. El valor de esta etiqueta debería contener una descripción razonable. Debemos evitar, por ello, abreviaciones a no ser que sea absolutamente necesario. Hay que destacar que esta información (válido para el resto de etiquetas) será probablemente utilizada por un programa software para realizar búsquedas. Si el nombre del evento es desconocido, se debe indicar el siguiente valor: "?". Ejemplos: • [Event "FIDE World Championship"] • [Event "Moscow City Championship"] • [Event "ACM North American Computer Championship"] • [Event "Casual Game"] • [Event "?"]
Site	Lugar donde el evento se llevó a cabo. El formato debe ser el siguiente (no olvidemos que debe estar encerrado todo el valor entre comillas dobles): • Ciudad • "," • Región • País: Siglas en mayúsculas, 3 dígitos de acuerdo con el código del Comité Olímpico Internacional. Por ejemplo: "México, D.F. MEX". Si el nombre del lugar es desconocido se debe indicar el valor con un "?". Ejemplos: • [Site "New York City, NY USA"] • [Site "St. Petersburg RUS"] • [Site "Riga LAT"] NOTA: *En uno de los anexos de este documento se incluyen todas las siglas de países del Comité Olímpico Internacional (COI)*.
Date	La fecha del inicio de la partida en formato AAAA.MM.DD (AAAA=Año, MM=mes, DD=día). Cuando se desconoce alguno de los valores se pone "??".

	Hay que resaltar que no tiene porqué coincidir la fecha del inicio de la partida con la fecha del inicio del evento donde se celebra la partida. La fecha siempre se indica respecto de la fecha local donde se celebra el evento. Ejemplos: • [Date "1992.08.31"] • [Date "1993.??.??"] • [Date "2001.01.01"] Como ya hemos visto anteriormente, existen combinaciones con el carácter "?" no válidas.
Round	La etiqueta "round" contiene la ronda en la que se realiza la partida. En una competición de tipo "match" se debe indicar la partida jugada. En ocasiones, la indicación de la ronda podría ser inapropiada. En estos casos, se debe indicar como ronda un guión (-). En cambio, si desconocemos la ronda, deberíamos indicarlo con el símbolo de interrogación (?). Ejemplos: • [Round "1"] • [Round "3.1"] • [Round "-"] • [Round "?"]
White	El jugador(es) de las piezas blancas, en el mismo formato que aparecería en una guía telefónica: Apellido, ",", espacio en blanco, y nombre. Si desconocemos el nombre del jugador, debemos incluir como valor un símbolo de interrogación (?). En algunos eventos no oficiales, un bando puede estar compuesto de más de un jugador. Para indicar esta circunstancia, los nombre de los jugadores deben ir separados por ":" y colocados por orden alfabético. Si el jugador de la partida es un programa informático, se debe indicar tras el nombre de dicho programa la versión del mismo. El formato utilizado por la FIDE es el mismo que hemos indicado para esta etiqueta. Obviamente, este comentario también es válido para el bando de las negras. Ejemplos: • [White "Cabeza, Guillermo:Pujó, José David:Herreros, Jorge Alejandro"] • [White "Tal, Mikhail N."] • [White "van der Wiel, Johan"] • [White "Acme Pawngrabber v.3.2"] • [White "Fine, R."] • [White "?"]
Black	El jugador(es) de las piezas negras, con el mismo formato que el utilizado para las piezas blancas. Ejemplos: • [Black "Lasker, Emmanuel"] • [Black "Smyslov, Vasily V."] • [Black "Smith, John Q.:Woodpusher 2000"] • [Black "Morphy"] • [Black "?"]
Result	El valor de la etiqueta "result" contiene el resultado de la partida. Siempre debe coincidir con el resultado indicado al final de los movimientos de una partida (sección 2). Sólo se pueden indicar 4 posibles valores: • "1-0" – El jugador con las piezas blancas ganó. (es un cero, no la letra "O" mayúscula). • "0-1" – El jugador con las piezas negras ganó. • "1/2-1/2" – Tablas (empate, medio punto para cada jugador). • "*" – El juego está actualmente en disputa, o ha sucedido algún acontecimiento extraordinario, o

	el resultado es desconocido. Ejemplos: • [Result "0-1"] • [Result "1-0"] • [Result "1/2-1/2"] • [Result "*"]

Tabla V: Etiquetas "STR".

En los ficheros con formato "export", además, se incluye una línea en blanco tras la última pareja "etiqueta-valor" y antes de la sección de movimientos de la partida. La idea inicial para realizarlo de esta forma fue para facilitar el trabajo de los programas de escaneo y tratamiento de la información.

Sección 2 partida PGN: movimientos de la partida

La sección de texto de movimientos está compuesta de movimientos de ajedrez, indicadores de número de movimiento, anotaciones (opcional) y un delimitador de resultado de partida. Dado que un movimiento ilegal no está permitido en una partida de ajedrez, tampoco está permitido incluirlos dentro de un fichero PGN.

Para anotar las jugadas de una partida se debe utiliza la denominada "notación estándar algebraica" (NEA en adelante). El texto de las jugadas en NEA describe los movimientos realizados por cada jugador. La estructura de movimientos es la siguiente:

- Número de movimiento. Ejemplo: 1. (el punto final forma parte de la notación)
- Movimiento de las blancas.
- Movimiento de las negras.

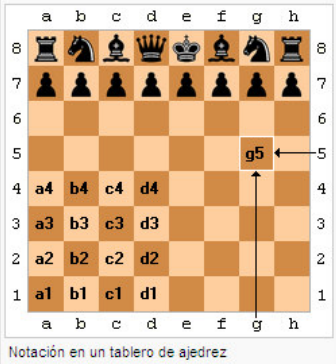


Figura XIV: notación de filas y columnas.

A su vez, un movimiento de cualquier bando se representa con la siguiente secuencia:

- Letra descriptiva de la pieza en inglés, aunque se puede indicar la pieza en otros idiomas.
- "x" si existe una captura de una pieza contraria.
- El código algebraico (dos caracteres) que indica la posición final (escaque) donde la pieza se desplazó.

En la figura adjunta podemos ver como se indican las distintas filas y columnas de un tablero de ajedrez. La fila más cercana a las piezas blancas al inicio de una partida es la fila "1", la siguiente es la fila "2" y así, sucesivamente, hasta llegar a la fila "8".

Del mismo modo, desde la posición del jugador de blancas, la primera columna de la izquierda es la columna "a", la siguiente la "b" y así, sucesivamente, hasta llegar a la columna "h". Obviamente, desde la perspectiva de las negras, las filas y columnas son al contrario, pero es importante resaltar

que siempre debemos marcar las filas y columnas de esta forma, aunque las piezas de las negras se encuentren en la parte inferior de la figura (perspectiva de las negras).

Las abreviaturas de las piezas para cada uno de los idiomas permitidos son las siguientes:

Idioma	Peón (Pawn)	Caballo (Knight)	Alfil (Bishop)	Torre (Rook)	Dama (Queen)	Rey (King)
Czech	P	J	S	V	D	K
Danish	B	S	L	T	D	K
Dutch	O	P	L	T	D	K
English	**P**	**N**	**B**	**R**	**Q**	**K**
Estonian	P	R	O	V	L	K
Finnish	P	R	L	T	D	K
French	P	C	F	T	D	R
German	B	S	L	T	D	K
Hungarian	G	H	F	B	V	K
Icelandic	P	R	B	H	D	K
Italian	P	C	A	T	D	R
Norwegian	B	S	L	T	D	K
Polish	P	S	G	W	H	K
Portuguese	P	C	B	T	D	R
Romanian	P	C	N	T	D	R
Spanish	**P**	**C**	**A**	**T**	**D**	**R**
Swedish	B	S	L	T	D	K

Tabla VI: Abreviaturas de piezas por idioma.

NOTA: *En NEA, al peón se le da una abreviatura vacía, es decir, no se indica la letra "P" y pasamos directamente a indicar la posición final de la pieza. Indicamos aquí su abreviatura porque en otros contextos sí se utiliza*.

En ocasiones, dos piezas del mismo tipo pueden trasladarse a la misma posición final. Por ejemplo, muy a menudo sucede que con dos torres que se encuentran en la primera fila, sin otras piezas entre ellas, deseamos mover una de ellas a otra posición dentro de la misma fila. Dado que ambas podrían ocupar dicha posición, necesitamos ampliar la notación para hacer referencia a una de ellas. Si ambas piezas se encuentran en la misma fila, haremos referencia a la columna de la pieza que movemos. Si ambas piezas se encuentran en la misma columna, haremos referencia a la fila de la pieza que movemos. Obviamente, dos piezas no pueden estar en la misma fila y columna, dado que estarían ocupando el mismo escaque. Siguiendo el ejemplo anterior podríamos indicar "cRd1" para indicar que la torre de la columna "c" se desplaza a la posición "d1". NEA también permite la indicación del escaque exacto donde se encuentra una pieza antes del movimiento, para evitar ambigüedades. El ejemplo anterior también se podría haber expresado como "c1Rd1" (la torre del escaque "c1" se desplaza a la posición "d1"), aunque esta variante de notación se utiliza menos.

El resto de posibles anotaciones de una partida son las siguientes:

- Enroque corto: "O-O". (importante, son "oes" mayúsculas, no ceros).
- Enroque largo: "O-O-O".
- Promoción de peón: se denota añadiendo un signo "=" seguido del nombre de la pieza a la que promociona el peón (Q, R, B, N en notación inglesa: dama, torre, alfil o caballo respectivamente).

- Jaque: Se añade un "+" al final del movimiento.
- Jaque mate: Se añade un "#" al final del movimiento.

NEA también acepta la inclusión de comentarios en el movimiento, aunque no pueden anidarse o solaparse:

- ";": comentario hasta el final de la línea.
- "{": inicio de comentario multi-línea.
- "}": fin de comentario multi-línea.

Veamos a continuación un par de ejemplos de partidas completas muy conocidas dentro del mundo del ajedrez:

```
[Event "Informal Game"]
[Site "London, England ENG"]
[Date "1851.07.??"]
[Round "-"]
[White "Anderssen, Adolf"]
[Black "Kieseritzky, Lionel"]
[Result "1-0"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10 g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19.e5 Qxa1+ 20.Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3
O-O 9. h3 Nb8 10. d4 Nbd7 11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15.
Nb1 h6 16. Bh4 c5 17. dxe5 Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21.
Nc4 Nxc4 22. Bxc4 Nb6 23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7
27. Qe3 Qg5 28. Qxg5 hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33.
f3 Bc8 34. Kf2 Bf5 35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5
40. Rd6 Kc5 41. Ra6 Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

Etiquetas complementarias

Existen muchos otros datos con los que podemos completar la información asociada a una partida, también incluido en el estándar PGN, mediante un conjunto de nuevas etiquetas. A continuación revisaremos todas estas etiquetas por grupos de significado.

Información adicional relacionada con los jugadores

En este bloque comentaremos las etiquetas que complementan la información de los jugadores que disputan una partida.

Etiqueta	Significado
WhiteTitle	Indica el título, de tenerlo, asociado al jugador. Ejemplos: • IM: Maestro Internacional. • GM: Gran Maestro. Como ya comentamos en su momento, una partida puede ser disputada por más de un jugador por bando. Para indicar los títulos de cada jugador se procederá de la misma forma, indicando cada título separado por ":". En el caso de que un jugador no tenga título reconocido, se pondrá el valor "-". Ejemplos: • [White "Cabeza, Guillermo:Pujó, José David:Herreros, Jorge Alejandro"] • [WhiteTitle "IM:-:GM"] Es decir, Guillermo Cabeza es Maestro Internacional, José David Pujó no tiene título reconocido por la FIDE, y Jorge Alejandro Herreros es Gran Maestro.
BlackTitle	El mismo significado que la etiqueta anterior, pero para el jugador de negras.
WhiteElo	El ELO (en esta ocasión no es un acrónimo, es el apellido de su inventor) es un indicador que marca la fuerza de un jugador. En España y en muchos otros países se obtiene obteniendo tres "bloques" en tres competiciones valederas para ELO (partidas de 2 horas por jugador, autorizadas por el organismo competente. En España, la Federación de Ajedrez – FEDA. La Federación Internacional, FIDE, en el caso de torneo valedero para ELO internacional). Un bloque se obtiene ganando o realizando tablas contra un jugador con ELO, y habiendo jugado, al menos, 3 partidas contra jugadores con ELO en ese mismo campeonato/torneo. Normalmente, un jugador que comienza su andadura por el mundo de ajedrez suele obtener ELOs FEDA entre los valores 1200 y 1300. El actual campeón del mundo, Viswanathan Anand, tiene un ELO FIDE de 2817 puntos. Si el jugador no tiene ELO se incluye el valor "-". Es importante resaltar que en esta etiqueta se indica el ELO FIDE, no el ELO nacional. Ejemplo: • [WhiteElo "1490"] NOTA: *En general, y para no repetir la misma información en cada etiqueta, el valor "-" en una etiqueta identifica que no se conoce el valor o no se tiene un valor asociado.*
BlackElo	El mismo significado que la etiqueta anterior, pero para el jugador de negras. Ejemplo: • [BlackElo "1212"]
WhiteUSCF	Valor entero para indicar el ELO del jugador de blancas otorgado por la USCF (United States Chess Federation – Federación de Ajedrez de los Estados Unidos). Otras etiquetas homólogas pueden ser utilizadas para identificar otras federaciones: FEDA, etc. Ejemplo: • [WhiteUSCF "1512"]
BlackUSCF	El mismo significado que la etiqueta anterior, pero para el jugador de negras. Ejemplo: [BlackUSCF "1888"]
WhiteNA	"Network Addresses". Correo electrónico o cualquier otra dirección de red del jugador con blancas. Ejemplo:

	• [WhiteNA "jlherreros@uoc.edu"]
BlackNA	El mismo significado que la etiqueta anterior, pero para el jugador de negras. Ejemplo: • [BlackNA "jlherreros@gmail.com"]
WhiteType	Para indicar el tipo de jugador de una partida. Sólo podemos incluir los siguientes valores: • "human": jugador humano. • "program": jugador no humano (ordenador, PSP, etc.). Ejemplos: • [WhiteType "human"] • [WhiteType "program"]
BlackType	El mismo significado que la etiqueta anterior, pero para el jugador de negras. Ejemplo: • [BlackType "program"]

Tabla VII: Etiquetas PGN sobre jugadores.

Información adicional relacionada con el evento

En este bloque comentaremos las etiquetas que complementan la información del evento donde se disputa una partida.

Etiqueta	Significado
EventDate	De igual formato que la etiqueta Date, pero para indicar la fecha de comienzo de un evento. Como se ha comentado, un evento puede durar más de un día, no coincidiendo en muchos casos el valor de esta etiqueta con el valor de la etiqueta "Date". Ejemplo: • [EventDate "2011.03.12"]
EventType	Tipo de evento (nueva etiqueta PGN).
EventCountry	Ciudad donde se realiza el evento (nueva etiqueta PGN).
EventRounds	Rondas del evento (nueva etiqueta PGN).
EventSponsor	Etiqueta con valor alfanumérico para indicar el patrocinado (sponsor) del evento. Ejemplo: • [EventSponsor "Universitat Oberta de Catalunya"]
Section	Etiqueta con valor alfanumérico para indicar la categoría del evento: "Open", "Reserve", etc. Ejemplo: • [Section "Open"]
Stage	Etiqueta con valor alfanumérico para indicar el momento concreto dentro de un evento: "Preliminary", "Semifinal", etc. Ejemplo:

	• [Stage “Final”]
Board	Etiqueta con valor numérico para indicar el tablero de juego, por ejemplo, para eventos por equipos o en exhibiciones de simultáneas. Ejemplo: • [Board “2”]

Tabla VIII: Etiquetas PGN sobre eventos.

Información adicional relacionada con la apertura realizada

En este bloque comentaremos las etiquetas que complementan la información de la apertura realizada en la partida. Los valores incluidos en estas etiquetas pueden variar en función del lenguaje local utilizado.

Etiqueta	Significado
Opening	Esta etiqueta se utiliza para indicar el nombre tradicional de la apertura realizada. Internamente se identifica con el nombre “OpCode v0”. Ejemplo: • [Opening “Caro-Kann”]
Variation	Esta etiqueta se utiliza para indicar el nombre tradicional de la variante de la apertura realizada. Internamente se identifica con el nombre “OpCode v1”. Ejemplo: • [Opening “variante del cambio”]
SubVariation	Esta etiqueta se utiliza para indicar el nombre tradicional de la sub-variante de la apertura realizada. Internamente se identifica con el nombre “OpCode v2”. Ejemplo: • [SubVariation “Variante Rubinstein”]
ECO	Código de apertura según la Encyclopedia of Chess Openings. Este código tiene el formato “XDD” o “XDD/DD” con el siguiente significado: • “X”: letra de la “A” a la “E”. • “DD”: dígitos numéricos, desde el “00” hasta el “99”. Internamente se identifica con el nombre “OpCode eco”. Ejemplo: • [ECO “B13”]
NIC	Código de apertura según la “New in Chess Database”. Internamente se identifica con el nombre “OpCode nic”. Ejemplo: • [ECO “SI 3.5”]

Tabla IX: Etiquetas PGN sobre la apertura realizada.

Información adicional relacionada con fechas y horas

En este bloque comentaremos las etiquetas que complementan la información relacionada con la hora de una partida y la fecha y hora de la misma en referencia con el huso horario UTC.

Etiqueta	Significado
Time	La hora local de comienzo de la partida en formato "HH:MM:SS" (HH=hora, MM=minutos, SS=segundos). Hay que resaltar que el carácter "dos puntos" (:) se utiliza como separador, en lugar del "punto" (.) utilizado en los valores de las etiquetas fecha. Por hora local se entiende la hora del lugar (etiqueta "Site") donde se realiza el evento. Ejemplo: • [Time "10:00:00"]
UTCTime	Esta etiqueta es similar a la anterior, pero para indicar en este caso la hora correspondiente al UTC (Universal Coordinated Time). Ejemplo: • [UTCTime "13:10:10"]
UTCDate	Esta etiqueta es similar a la etiqueta "Date", pero para indicar en este caso la fecha correspondiente UTC (Universal Coordinated Time). En ocasiones, podría ser distinta en 1 día con la etiqueta "Date". Ejemplo: • [UTCDate "2011.03.22"]

Tabla X: Etiquetas PGN de fechas y horas.

Información adicional relacionada con el control de tiempos

En este bloque comentaremos la etiqueta que complementa la información relacionada con el método empleado para controlar los tiempos de una partida.

Etiqueta	Significado
TimeControl	Esta etiqueta permite listar uno o más campos de control de tiempos. Cada campo contiene un descriptor para cada período de control de tiempo. Para separar cada uno de los campos debemos utilizar el carácter "dos puntos" (:). Lo descriptores deben aparecer en el mismo orden en el que serán utilizados en la partida. El último descriptor indicado en esta etiqueta será el que deba utilizarse (repetirse) en el caso de que se requieran más períodos de tiempo. Hay 6 tipos de campos de control de tiempos: • El primer tipo es un simple carácter "interrogación" (?), el cual indica que se desconoce el control de tiempos utilizado en la partida. Cuando se utiliza este descriptor, suele ser de manera aislada, es decir, sin utilizar otros descriptores en este mismo campo. • El segundo tipo de descriptor es el carácter "guión" (-), el cual indica que no se ha utilizado ningún tipo de control de tiempos en la partida. Cuando se utiliza, obviamente, es el único descriptor incluido en el campo. • El tercer tipo de descriptor está formado por dos enteros separados por un carácter "barra inclinada" (/). El primer entero es el número de movimientos en el período y el segundo entero es el número de segundos permitidos en el período. Por ejemplo, si queremos indicar

	que se deben realizar 40 movimientos en un período de 2 horas y media, debemos indicarlo como “40/9000”. • El cuarto tipo de descriptor se utiliza para indicar el período de control “muerte súbita”. Debería ser utilizado siempre como último descriptor del campo. En algunas ocasiones, es el único descriptor presente. El formato de este descriptor es un único valor entero que indica el número de segundos del período. Por ejemplo, en una partida de modalidad blitz (5 minutos por jugador), el valor indicado sería “300”. • El quinto tipo de descriptor se utiliza para indicar un período de tiempo incremental. Este descriptor debería ser utilizado como último descriptor y debería ser además único. El formato del campo está compuesto de dos números enteros separados por el carácter “más” (+). El primer entero indica el número de segundos contemplados en el período de tiempo, y el segundo entero representa el número de segundos de incremento por cada movimiento. Por ejemplo, una partida de 90 minutos más un extra de un minuto por movimiento estaría representado por “4500+60”. • El sexto y último tipo de descriptor se utiliza para los controles de tiempo del tipo “reloj de arena”. Debe ser utilizado como último descriptor de la cadena y, en ocasiones, debería ser el único descriptor utilizado en el campo. El formato del campo está compuesto de un carácter “asterisco” (*) seguido por un entero positivo. El valor entero indica el número de segundos en esta modalidad (tiempo hasta el final) para cada jugador. Por ejemplo, para una partida de 3 minutos, el valor de este campo sería “*180”. Según la especificación PGN, otros descriptores podrán ser utilizados en el futuro si fuese necesario.

Tabla XI: Etiquetas PGN de control de tiempos.

Información adicional relacionada con posiciones de inicio alternativas

En este bloque comentaremos las etiquetas utilizadas cuando el inicio de una partida se realiza en una posición distinta a la habitual. Se pueden utilizar estas etiquetas para indicar, por ejemplo, posiciones que representan ejercicios a resolver, objetivo principal de este PFC. También se puede utilizar para indicar inicios de partida no convencionales como, por ejemplo, la “variante alternativa de Fisher”.

Etiqueta	Significado
SetUp	Esta etiqueta se utiliza para indicar cómo se ha iniciado la partida. Con el valor entero “0” se indica que la partida ha comenzado con todas las piezas y peones en sus posiciones de inicio “legales”, según el ajedrez tradicional descrito en las normas FIDE. El valor “1” indica que la posición inicial no es la habitual, por lo que habrá que utilizar en paralelo la etiqueta “FEN” (se verá a continuación) para determinar la posición de todas las piezas. Ejemplo: • [SetUp “1”]
FEN	Representa la posición inicial del juego en notación “Forsyth-Edwards Notation” (FEN). Se utiliza para representar juegos parciales que empiezan en alguna posición determinada, o para indicar otras variantes de inicio, como sucede en el juego de ajedrez aleatorio de Fischer, donde la posición inicial no coincide con la del ajedrez convencional. También se puede utilizar para indicar un inicio donde uno de los jugadores “regala” una pieza a su oponente, circunstancia que ocurre muy a menudo en partidas de exhibición. Si se utiliza una etiqueta de tipo “FEN”, es obligado el uso de la etiqueta “SetUp” (inicialización) con el valor “1” indicado. Dado que nuestro proyecto trata de resolver posiciones de ajedrez, a modo de ejercicios, se utilizará

	básicamente este formato, indicando en la sección de jugadas los movimientos más adecuados, a modo de solución del ejercicio propuesto. NOTA: *En el siguiente apartado podremos ver el análisis completo del contenido del formato FEN*. Ejemplo: • [FEN "r1bqk2r/pppn1ppp/5n2/2B5/2B1p3/2N5/PPP1Q1PP/R4RK1 w kq - 0 1"]

Tabla XII: Etiquetas PGN de inicio anternativo de partidas.

Información adicional relacionada con la terminación de una partida

En este bloque comentaremos la etiqueta que complementa la información relacionada con el modo en el que ha finalizado una partida, independientemente del resultado obtenido.

Etiqueta	Significado
Termination	Esta etiqueta presenta las razones o matices de la conclusión de una partida, dado que la etiqueta "result", por sí sola, no nos permite conocer los detalles de cómo se ha obtenido el resultado. Los valores posibles son los siguientes: • "abandoned": Abandono: El jugador que ha perdido el encuentro ha abandonado. • "adjudication": Adjudicación: El resultado ha sido determinado por adjudicación de una tercera parte, normalmente, el árbitro. • "death": Muerte del jugador que pierde la partida. En este punto mantendré el texto en inglés incluido en el fichero oficial de normalización PGN: "*losing player called to greater things, one hopes*." (¡Todos lo esperamos!). • "emergecy": Emergencia. Juego concluido debido a circunstancias imprevistas. • "normal": Normal, el juego termina de forma normal. • "rules infraction": Infracción a las reglas. Un observador (árbitro de la contienda) detecta y pone de manifiesto el incumplimiento grave a alguna de las reglas de juego (FEDA o FIDE), o del evento en particular. • "time forfeit": Tiempo acabado. Pérdida de partida debida a la falta de tiempo en el reloj que controla el ritmo de juego. • "unterminated": No finalizado. El juego no ha terminado todavía, pero ya se ha realizado una representación de la partida en formato PGN. Ejemplo: • [Termination ""abandoned""]

Tabla XIII: Etiquetas PGN de finalización de partida.

Información adicional relacionada con otros datos de interés

En este bloque comentaremos las etiquetas que complementa algún dato más de una partida de ajedrez, a modo de "miscelánea".

Etiqueta	Significado
Annotator	Esta etiqueta se utiliza para indicar el nombre, o nombres, de las personas que han realizado comentarios (anotaciones) de la partida en cuestión. Su formato es el mismo que el ya visto para el nombre del jugador, o jugadores, de blancas. Ejemplo: • [Annotator “Rudolf, Anna”]
Mode	Esta etiqueta se utiliza para indicar cómo (entorno físico) se ha realizado la partida. Algunos valores posibles son los siguientes: • “OTB” – Sobre el tablero (Over the board). • “PM” – Vía correo tradicional (paper mail). • “EM” – Vía correo electrónico (electronic mail). • “ICS” – Servidor de ajedrez vía Internet (Internet Chess Server). • “TC” – Vía telecomunicación genérica (general telecommunication). Ejemplo: • [Mode “ICS”]
PlyCount	Esta etiqueta se utiliza para indicar el número de movimientos (ply) ya realizados en una partida contemplando todos los movimientos indicados. Ejemplo: [PlyCount “56”]

Tabla XIV: Etiquetas PGN complementarias.

1.2.3 Formato de representación de posiciones FEN

En esta sección presentamos la notación FEN, necesaria para poder interpretar como se anota una posición de una partida de ajedrez dentro de los ficheros PGN que trataremos en nuestro generador.

La notación de Forsyth-Edwards (del inglés Forsyth-Edwards Notation, FEN en adelante) se utiliza para registrar una posición determinada de una partida de ajedrez. En la segunda sección de una partida de ajedrez en formato PGN, como hemos visto, se indican todos los movimientos realizados desde un momento determinado, el cual puede ser el inicio de la partida o cualquier otro punto de la misma. Para poder indicar una posición distinta a la inicial se utiliza la notación FEN. Esta notación se utiliza, por lo tanto, para:

- Variantes del ajedrez como en el Ajedrez aleatorio de Fischer, donde la posición inicial no es la del ajedrez convencional.
- En los casos de problemas de ajedrez (nuestro principal objetivo), o en soluciones artísticas.

A continuación destacaremos otras situaciones en las que se utiliza este formato:

- A nivel de desarrollo informático es muy conocida esta notación, siendo la utilizada en los aplicativos actuales, junto con la notación PGN.
- En los sitios de internet también se utiliza cuando no se pueden utilizar gráficos. Muchos jugadores experimentados son capaces de reproducir una posición en la mente simplemente leyendo este contenido.
- Bases de datos (.dat) de aperturas y finales.

Esta notación está compuesta de los siguientes elementos:

- Campo 1: Posición de las piezas.
- Campo 2: Turno de juego.
- Campo 3: Posibilidades de enroque.
- Campo 4: Posibilidad de “tomar al paso”.
- Campo 5: Movimientos sin captura.
- Campo 6: Movimientos de partida.

Estos campos están separados entre sí por un carácter “espacio en blanco”, lo cual significa que dentro de cada campo no está permitido utilizar dicho carácter.

Campo 1: Posición de las piezas

Para indicar una posición determinada de una partida de ajedrez se utiliza un string con el siguiente formato: “fila8/fila7/fila6/fila5/fila4/fila3/fila2/fila1/”, donde “fila8” indica la ubicación de piezas sobre la fila 8, de izquierda a derecha y así, sucesivamente, hasta indicar las piezas ubicadas en la fila 1, también de izquierda a derecha. Ya hemos visto, en capítulos anteriores, cómo se

representaban habitualmente las filas y las columnas de un tablero de ajedrez (filas de la "1" a la "8", columnas de la "A" a la "H"), siendo la "fila 8" la fila más alejada a la posición del jugador de blancas. Las piezas blancas se indican con una letra en mayúsculas, mientras que las piezas negras se indican con una letra en minúsculas. También hemos analizado anteriormente las letras utilizadas para representar cada pieza del tablero, siendo dependientes del idioma utilizado. Por ejemplo, en el caso del idioma inglés, las letras a utilizar serían las siguientes:

Piezas	Peón (Pawn)	Caballo (Knight)	Alfil (Bishop)	Torre (Rook)	Dama (queen)	Rey (King)
Blancas	P	N	B	R	Q	K
Negras	p	n	b	r	q	k

Tabla XV: Notación FEN, identificación de piezas.

NOTA: *En los siguientes apartados utilizaremos las letras que representan las piezas en el idioma español, para clarificar los ejemplos*.

Se anotará, escaque a escaque, la inicial de la pieza que este en cada posición, con mayúsculas para blancas y minúsculas para negras. En el caso de que haya espacios sin piezas se anotará el número de espacios entre pieza y pieza.

Para separar una fila de otra es necesario poner una "barra diagonal" (/). Veamos un ejemplo que representa la posición inicial de una partida de ajedrez convencional:

tcadract/pppppppp/8/8/8/8/PPPPPPPP/TCADRACT

Podemos ver que las cuatro filas centrales no tienen piezas en momento inicial de la partida, por ello el valor "8" en cada fila.

Campo 2: turno de juego

Al terminar de anotar todas las piezas en su lugar, debemos indicar las especificaciones técnicas del juego (todos los campos separados por un espacio en blanco). En primer lugar indicamos la inicial del bando que tiene el turno:

- "b" o "B": las blancas mueven. En inglés sería "w" o "W".
- "n" o "N": las negras mueven. En inglés sería "b" o "B".

Ejemplo, les toca mover a las blancas:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B

Campo 3: posibilidades de enroque

Dejamos, de nuevo, un espacio en blanco e indicamos las posibilidades de enroque por ambos bandos. Para blancas, se indica "R" si hay posibilidad de enroque del lado del rey, y "D" si hay posibilidad de enroque del lado de la dama; y lo mismo para el bando negro pero con minúsculas.

Ejemplo: Las blancas tienen aún posibilidad de enrocar en el flanco de dama, y las negras en el flanco de rey:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B Dr

Campo 4: posibilidad de "tomar al paso"

En las reglas del ajedrez, cuando un peón avanza dos posiciones desde su posición inicial, puede ser capturado por otro peón que se encuentre en una casilla de columnas colindantes (a la derecha o izquierda) de la posición final de dicho peón (en la misma fila final). Para poder indicar esta circunstancia se utiliza este campo. Tras el espacio en blanco, anotaremos sólo si en la última jugada se movió un peón dos casillas, indicando en este caso la casilla por la que paso. Por ejemplo, en la apertura "1. e4" anotaríamos la casilla "e3", que fue por donde paso el peón. En el caso de que no se pueda tomar al paso porque un peón no se haya movido dos casillas, solamente pondremos en este campo un carácter "guión" (-).

Ejemplo: Las blancas pueden tomar al paso en f6:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B Dr f6

(el último movimiento realizado por las negras fue el peón, de f7 a f5)

Ejemplo: Las blancas no pueden tomar al paso:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B Dr -

Campo 5: movimientos sin captura

Dejamos otro espacio en blanco y ponemos el número de "medios movimientos" de cada jugador que han transcurrido desde la última captura o el último movimiento de peón. Un "movimiento" en ajedrez contempla una jugada de blancas y una jugada de negras. Cada uno de los movimientos de cada jugador se considera "medios-movimientos", y es el valor que nos interesa para este campo.

Esta información es necesaria para poder conocer los medio-movimientos restantes para alcanzar unas tablas, circunstancia que se alcanza cuando se realizan 50 medio-movimientos bajo estas condiciones.

Ejemplo: Se han realizado 12 movimientos de piezas sin captura y sin ningún movimiento de peón:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B Dr - 12

Campo 6: movimientos de partida

Por último, y tras dejar un nuevo espacio en blanco, se indica el número de movimientos totales (en este caso, no son "medio-movimientos" de la partida).

Ejemplo, se han realizado 56 medio-movimientos de blancas y 55 medio-movimientos de negras:

fila8/fila7/fila6/fila5/fila4/fila3/fia2/fila1/ B Dr – 12 56

Veamos algunos ejemplos completos, con la descripción asociada a cada ejemplo, y donde podemos ver cómo se va actualizando este sexto campo:

La notación FEN para la posición inicial:

tcadract/pppppppp/8/8/8/8/PPPPPPPP/TCADRACT b RDrd - 0 1

Tras el movimiento de blancas "1. e4":

tcadract/pppppppp/8/8/4P3/8/PPPP1PPP/TCADRACT n RDrd e3 0 1

Tras un nuevo movimiento, de negras en esta ocasión, "1. ... c5":

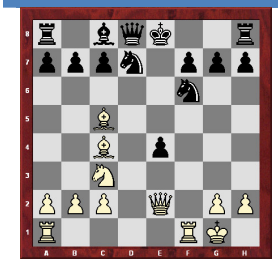
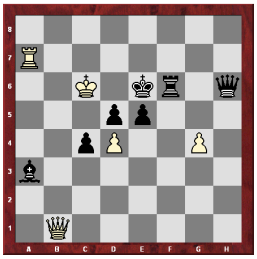
tcadract/pp1ppppp/8/2p5/4P3/8/PPPP1PPP/TCADRACT b RDrd c6 0 2

Un nuevo movimiento de blancas, "2. Cf3":

tcadract/pp1ppppp/8/2p5/4P3/5C2/PPPP1PPP/TCADRA1T n RDrd - 1 2

Podemos ver en los ejemplos anteriores, que podrían tratarse de los cuatro primeros movimientos de una partida, cómo se alterna el turno de juego, empezando el bando blanco ("b") y moviendo a continuación el bando negro ("n"). En los cuatro primeros movimientos están disponibles todos los enroques ("RDrd") aunque esto no significa que se puedan realizar en estos momentos, dado que hay piezas en el camino del rey que impiden dicha acción. En el primer movimiento de blancas y primer movimiento de negras se podría capturar al paso (en "e3" y "c6"), si hubiese un peón contrario en una ubicación adecuada para ello. En el resto de posiciones no es posible capturara al paso ("-"). En el último movimiento se indica un "1" para denotar los medios-movimientos sin captura o movimiento de peón, y en el último campo vemos los movimientos totales que se han jugado: "1", "1", "2", "2". Los siguientes serán: "3", "3", "4", "4", etc.

Para cerrar el análisis de la notación FEN presentaremos algunos ejemplos gráficos que aclaran los conceptos expuestos en este apartado (la notación de piezas está en inglés):

Figura	Posición en notación FEN
	r1bqk2r/pppn1ppp/5n2/2B5/2B1p3/2N5/PPP1Q1PP/R4RK1 w kq - 0 1
	8/R7/2K1kr1q/3pp3/2pP2P1/b7/8/1Q6 w - - 0 1

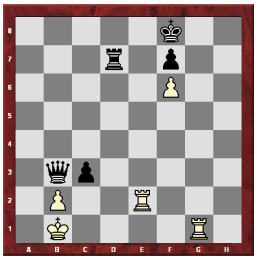
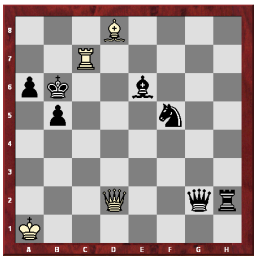
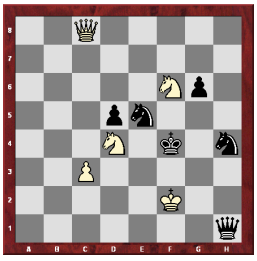
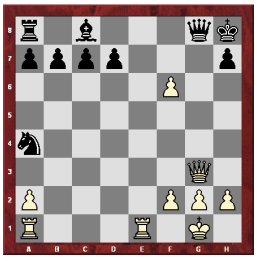
	5k2/3r1p2/5P2/8/8/1qp5/1P2R3/1K4R1 w - - 0 1
	r4b1r/pppbkBpp/q1n3n1/5p2/2NP4/1QP5/PP3PPP/RNB2RK1 w - - 0 1
	3B4/2R5/pk2b3/1p3n2/8/8/3Q2qr/K7 w - - 0 1
	6k1/1p1q2rp/p1bPRB2/8/1nP5/6Q1/PP4PP/7K w - - 0 1
	2Q5/8/5Np1/3pn3/3N1k1n/2P5/5K2/7q w - - 0 1
	r1b3qk/pppp3p/5P2/8/n7/6Q1/P4PPP/R3R1K1 w - - 0 1

Tabla XVI: Ejemplos de etiquetas FEN.

1.2.4 Ontología y el lenguaje OWL

Antes de desarrollar esta sección, vamos a responder algunas preguntas que nos permitirán entender los objetivos concretos de esta sección del documento y, en consecuencia, del PFC.

¿Qué es una ontología?

La literatura de inteligencia artificial contiene varias definiciones de ontología. Para los propósitos de este PFC, definimos una ontología como una descripción explicita y formal de conceptos en un dominio de discurso, mediante los siguientes elementos:

- Clases, a veces también denominadas "conceptos",
- Propiedades, que describen las características o atributos de la clase. En ocasiones, también denominadas "slots" o "roles".
- Restricciones, también denominados "facetas" o "restricciones de role", que incluyen condiciones adicionales a la información asociada a una propiedad.

NOTA: *En este documento se utilizará cualquiera de los términos anteriores indistintamente*.

Una "*base de conocimiento*" contiene una ontología y un conjunto de individuos de las clases que conforman la ontología. En realidad, hay una línea muy delgada donde la ontología termina y la base de conocimiento empieza. Precisamente la creación de esta base de conocimiento es uno de los objetivos principales del PFC. La ontología será definida en esta sección y posteriores, y las instancias serán creadas con el generador que desarrollaremos en el siguiente apartado. Nuestra base de conocimiento estará compuesta, por lo tanto, por contenido estático (clases y propiedades) y por contenido creado dinámicamente (individuos).

Las clases son el corazón de la mayoría de las ontologías, dado que describen los conceptos de un dominio. Por ejemplo, una clase de "posiciones de ajedrez" representa cualquier posible posición de una partida de ajedrez, desde la inicial con todas las piezas en su posición de origen hasta cualquier final con *jaque mate* posible. Una posición de ajedrez concreta, y toda su información asociada, serán en nuestra ontología instancias.

Una clase puede contener subclases que representan conceptos que son más específicos que la superclase. Por ejemplo, podríamos dividir la clase "posición" en posiciones más específicas con características particulares. Por ejemplo:

- Se podrían crear subclases para representar posiciones de una apertura de juego, posiciones de medio juego y posiciones de finales.
- Se podría dividir la clase "posición" en posiciones con posibilidad de ganancia de material, mejora de posición, posiciones de mate en 1 movimiento, 2 movimientos, etc.
- También sería posible una división con posiciones "legales" (contempladas en el reglamento) y posiciones ilegales, aunque esta división tendría poco interés instructivo.

Los slots, como se ha indicado, describen propiedades de clases y, en consecuencia, de las instancias de dichas clases. Por ejemplo, la posición conocida como “gambito de dama”, instancia de la clase “posición” podría tener dos propiedades:

- El primer slot define la posición de las piezas (todas las piezas en su posición de origen excepto el peón “d2” que ocupa la posición “d4”, el peón “c2” que ocupa la posición “c4” y el peón “d7” que ocupa la posición “d5”.
- El segundo slot define el jugador “en juego” (a quien le toca mover). En nuestro ejemplo, al jugador con piezas negras.

A nivel de clase, podríamos decir con la instancia anterior tendría dos slots que describen la posición de las piezas y el jugador en juego.

Por último, necesitaríamos indicar las restricciones de estos slots. Esta información adicional se consigue con las facetas.

En términos prácticos, desarrollar una ontología incluye:

- Definir clases en la ontología,
- Organizar las clases en una jerarquía taxonómica (subclase-superclase),
- Definir slots y describir valores permitidos para esos slots (facetas).
- Poblar los valores de los slots con las instancias.

Podemos, por todo ello, crear una base de conocimientos definiendo las instancias individuales de las clases definidas, precisando los valores específicos de los slots y restricciones adicionales sobre dichos slots.

¿Una ontología es como la conceptualización de un modelo en la programación orientada a objetos?

El desarrollo de ontologías es diferente al diseño de clases de la programación orientada a objetos (POO en adelante). La POO se centra principalmente en los métodos de una clase, tomando decisiones de diseño basadas en las propiedades operacionales de una clase; mientras que un diseñador de ontologías se toma esas decisiones basándonos en las propiedades estructurales de una clase. En consecuencia, la estructura de una clase y las relaciones entre clases en una ontología son diferentes de la estructura para un dominio similar en un programa orientado a objetos.

¿Por qué desarrollar una ontología?

En los últimos años, el desarrollo de ontologías se ha trasladado del entorno de los laboratorios de Inteligencia Artificial a los escritorios de los expertos de un dominio determinado. Las ontologías ya han llegado a ser comunes en Internet, presentes en grandes taxonomías que permiten categorizar sitios Web, como *Yahoo!*, o categorizar grandes librerías como *Amazon*.

El Consorcio *W3C* desarrolló el *Resource Description Framework* (RDF en adelante), un lenguaje para codificar conocimiento en páginas Web para hacerlas entendibles a los agentes no-humanos (procesos) que buscaban información. La Agencia de Proyectos de Investigación Avanzada en Defensa

(*Defense Advanced Research Projects Agency´- DARPA*), conjuntamente con el *W3C*, desarrolló "*DARPA Agent Markup Language*" (DAML), extensión del RDF con construcciones más expresivas para facilitar la interacción entre agentes en la Web.

En la actualidad, muchas disciplinas desarrollan sus propias ontologías estandarizadas para que los expertos de ciertos dominios puedan usarlas para compartir y anotar información en sus campos de trabajo. A modo de ejemplo, el campo de la medicina ha sido una de las áreas más desarrolladas en este sentido, generándose vocabularios de gran tamaño, con ánimo de estandarización, como:

- Snomed:
  - http://en.wikipedia.org/wiki/SNOMED_CT
- Unified Medical Language System:
  - http://en.wikipedia.org/wiki/Unified_Medical_Language_System

A partir de estas ontologías de gran repercusión, han surgido nuevas ontologías de propósito general. Por ejemplo, el programa de desarrollo de las naciones unidas (*United Nations Development Program*) junto con *Dun&Bradstreet* desarrollaron la ontología UNSPSC que provee toda la terminología requerida para productos y servicios: www.unspsc.org .

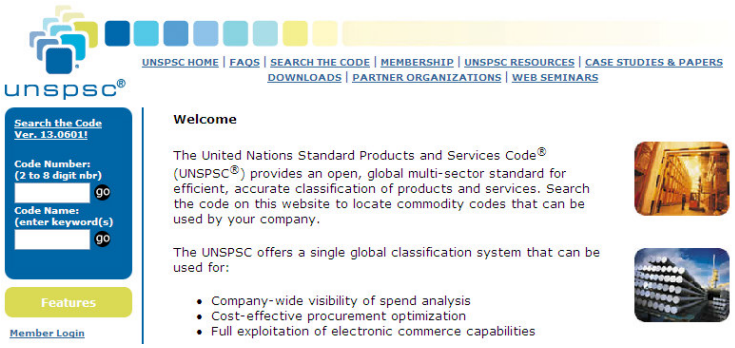


Figura XV: página web de UNSPSC.

Otras muchas ontologías pueden ser consultadas en la web *Ontolingua*:

http://www.ksl.stanford.edu/software/ontolingua/

En este apartado analizaremos las principales características del lenguaje OWL, conocimiento necesario para poder afrontar la definición de una ontología que describa el dominio de una posición de ajedrez, así como el desarrollo de individuos de la ontología que están relacionados con la representación gráfica de cada posición.

Introducción OWL

La mayoría de las páginas web actuales, ya sean estáticas o dinámicas, están concebidas para ofrecer información que será consultada por humanos, sin prestar atención a si dicha información puede ser procesada por una aplicación. El Lenguaje de Ontologías Web (OWL en adelante) está diseñado para

poder ser usado por aplicaciones que necesitan procesar el contenido de estas páginas. OWL ofrece mejores mecanismos de interpretabilidad de contenido web que los mecanismos anteriores, como los iniciales documentos XML, o los documentos RDF y RDF Schema (RDF-S), proporcionando vocabulario adicional junto a una semántica formal. Una ontología describe los términos de un dominio concreto y las relaciones entre dichos términos.

Como veremos a continuación, OWL ofrece tres sub-lenguajes con un nivel de expresividad diferente:

- OWL Lite.
- OWL DL.
- OWL Full.

El lenguaje OWL está descrito por la W3C por una serie de documentos, cada uno de ellos con una finalidad diferente, y orientados a una audiencia diferente. Estos documentos son los siguientes:

- Vista general de OWL: http://www.w3.org/TR/owl-features/ . Este documento ofrece una visión general de OWL, proporcionando una lista de las principales características del lenguaje, con breves descripciones.
- Guía OWL: http://www.w3.org/TR/owl-guide/ . Describe el uso del lenguaje OWL mediante un extenso ejemplo. También incluye un glosario con la terminología utilizada en todos los documentos OWL. Gran parte del contenido de este apartado ha sido consulado en esta guía.
- Referencia de OWL: http://www.w3.org/TR/owl-ref/ . Ofrece una descripción sistemática y compacta de todas las primitivas del modelado con OWL.
- Semántica y sintaxis abstracta de OWL: http://www.w3.org/TR/owl-semantics/ . Contiene la definición normativa final y formal del lenguaje OWL.
- Casos de prueba: http://www.w3.org/TR/owl-test/ . Contiene un amplio conjunto de casos de prueba para el lenguaje.
- Casos de uso y requerimientos OWL: http://www.w3.org/TR/webont-req/ . Contiene un conjunto de casos de uso para un lenguaje de ontologías web, y recopila un conjunto de requerimientos para OWL.

OWL y RDF / RDF Schema

La Web semántica es una visión de un futuro cada vez más cercano de la Web, donde la información está dando un significado explícito, permitiendo que las máquinas puedan procesarla automáticamente, e integrar dicha información disponible en la Web. Las bases de la web semántica son la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF para representar datos. El primer nivel requerido por encima de RDF para la web semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos Web. Si se espera que las máquinas hagan tareas útiles de razonamiento sobre estos documentos, el lenguaje debe ir más allá de las semánticas básicas del RDF Schema.

OWL ha sido diseñado, por ello, para cubrir esta necesidad de un lenguaje de ontologías web. OWL forma parte de un conjunto creciente de recomendaciones del W3C relacionadas con la Web semántica. A saber:

- XML: proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.

- XML Schema: es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos.
- RDF: es un modelo de datos para objetos ("recursos") y las relaciones entre ellos, proporcionando una semántica simple. Este tipo de modelo de datos puede ser representado en una sintaxis XML.
- RDF Schema: es un vocabulario utilizado para describir propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización, tanto de propiedades como de clases.
- OWL: añade más vocabulario para describir propiedades y clases. Entre otros:
    - relaciones entre clases (por ejemplo, desunión).
    - Cardinalidad (por ejemplo, "exactamente uno").
    - Igualdad entre clases.
    - Nuevos tipos de propiedades.
    - Capacidad de características de propiedades, como la simetría.
    - Posibilidad de clases enumeradas.

Sub-lenguajes OWL

OWL proporciona tres sub-lenguajes, como ya hemos indicado, cada uno de ellos con un nivel de expresividad mayor que el anterior, diseñados para ser usados por comunidades específicas de desarrolladores y usuarios:

- OWL Lite está diseñado para aquellos usuarios que necesitan principalmente una clasificación jerárquica y restricciones simples. Por ejemplo, a la vez que admite restricciones de cardinalidad, sólo permite establecer valores cardinales de 0 ó 1. Debería ser más sencillo proporcionar herramientas de soporte a OWL Lite que a las versiones con mayor nivel de expresividad, y OWL Lite proporciona una ruta rápida de migración para “tesauros” y otras taxonomías. OWL Lite tiene también una menor complejidad formal que el resto de sub-lenguajes.
- OWL DL está diseñado para aquellos usuarios que quieren la máxima expresividad conservando completitud computacional (se garantiza que todas las conclusiones sean computables), y resoluble (todos los cálculos se resolverán en un tiempo finito). OWL DL incluye todas las construcciones del lenguaje de OWL, pero sólo pueden ser usados bajo ciertas restricciones (por ejemplo, mientras una clase puede ser una subclase de otras muchas clases, una clase no puede ser una instancia de otra). OWL DL es denominado de esta forma debido a su correspondencia con la lógica de descripción (Description Logics, en inglés), un campo de investigación que estudia la lógica que compone la base formal de OWL.
- OWL Full está dirigido a usuarios que quieren máxima expresividad y mayor libertad sintáctica que con RDF, pero sin garantías computacionales. Por ejemplo, en OWL Full una clase puede ser considerada simultáneamente como una colección de clases individuales y como una clase individual propiamente dicha. OWL Full permite una ontología para aumentar el significado del vocabulario preestablecido (RDF ó OWL). Es poco probable que cualquier software de razonamiento sea capaz de obtener un razonamiento completo para cada característica de OWL Full.

Cada uno de estos sub-lenguajes es una extensión de su predecesor más simple, respecto a lo que puede ser expresado legamente y a la validación de sus conclusiones. El siguiente grupo de relaciones se mantienen, pero no las relaciones inversas:

- Cada ontología legal de OWL Lite es una ontología legal de OWL DL.
- Cada ontología legal de OWL DL es una ontología legal de OWL Full.
- Cada conclusión válida de OWL Lite es una conclusión válida de OWL DL.
- Cada conclusión válida de OWL DL es una conclusión válida de OWL Full.

Los desarrolladores de ontologías que adoptan OWL deberían considerar cuál es el sub-lenguaje que mejor se adapta a sus necesidades. La elección entre OWL Lite y OWL DL depende de las necesidades de los usuarios sobre la expresividad de las construcciones, proporcionando OWL DL las más expresivas. La elección entre OWL DL y OWL Full depende principalmente de las necesidades de los usuarios sobre los recursos de meta-modelado del esquema RDF (por ejemplo, definir clases de clases, o definir propiedades de clases). Cuando se usa OWL Full en comparación con OWL DL, el soporte en el razonamiento es menos predecible, ya que no existen en este momento implementaciones completas de OWL Full.

OWL Full puede ser considerada como una extensión de RDF, mientras que OWL Lite y OWL DL pueden ser consideradas como extensiones de una visión restringida de RDF. Cada documento OWL (Lite, DL, Full) es un documento RDF, y cada documento RDF es un documento de OWL Full, pero sólo algunos documentos RDF serán legalmente documentos OWL Lite ó OWL DL. Por este motivo, se ha de tener cuidado cuando un usuario quiera migrar un documento de RDF a OWL. Cuando se considere que la expresividad de OWL DL ó OWL Lite es adecuada, han de tomarse algunas precauciones para asegurar que el documento RDF original cumple con las restricciones adicionales impuestas por OWL DL y OWL Lite. Por ejemplo, cuando una URI es utilizada como nombre de una clase, debe indicarse explícitamente que esta URI debe ser una clase del tipo owl:Class (al igual que para las propiedades). Cada individuo debe estar definido como perteneciente, al menos, a una clase (incluso sólo con objetos owl:Thing); y las URIs usadas para las clases, propiedades e individuos deben ser disjuntos entre ellos.

Descripción del lenguaje OWL Lite

En esta sección presentamos un índice rápido de todas las características de OWL Lite, dado que será el sub-lenguaje utilizado en la definición de la ontología. OWL DL y OWL Full, por lo tanto, no serán detallados.

Algunos comentarios antes de describir el sub-lenguaje:

- los términos entre comillas son términos pertenecientes al lenguaje OWL, aunque también podrán identificar nombres de clases particulares. Ejemplos:
  - “subClassOf”.
  - “Posición”
- Los prefijos "rdf:" o "rdfs:" serán utilizados cuando los términos están ya presentes en RDF o en RDF Schema respectivamente. De lo contrario, los términos serán introducidos por “owl”, o sin prefijo. Ejemplos:

- "rdfs:subPropertyOf": indica que "subPropertyOf" ya se encuentra dentro del vocabulario "rdfs" (técnicamente: el rdfs namespace o espacio de nombres).
- "Class": está declarado de forma más precisa como owl:Class, y es un término introducido por OWL.

Construcciones del lenguaje OWL Lite

En OWL Lite, las clases sólo pueden ser definidas en términos de superclases (las superclases no pueden ser expresiones arbitrarias), y sólo pueden ser utilizados ciertos tipos de restricciones de clase. Además, únicamente se permite la equivalencia entre clases y las relaciones de subclases entre clases cuando se trata de clases definidas, pero no en el caso de expresiones de clases arbitrarias. Igualmente, las restricciones en OWL Lite usan sólo clases definidas. OWL Lite tiene, además, una noción limitada de cardinalidad, dado que las únicas cardinalidades que se pueden definir explícitamente son "0" ó "1".

Características del esquema RDF

Las siguientes características de OWL Lite están relacionadas con el esquema RDF.

Elemento	Significado
owl:Class (Thing, Nothing)	Una clase define un grupo de individuos que pertenecen a la misma porque comparten algunas propiedades. Por ejemplo, "Deborah" y "Frank" son miembros o individuos de la clase "Persona". Las clases pueden organizarse en una jerarquía de especialización usando "subClassOf". Clase general "Thing": Existe una clase general llamada "Thing" que es una clase de todos los individuos y es una superclase de todas las clases de OWL. Clase general "Nothing": También existe una clase general llamada "Nothing" que es la clase que no tiene instancias y es una subclase de todas las clases de OWL.
rdfs:subClassOf	Las jerarquías de clase deben crearse haciendo una o más indicaciones de que una clase es subclase de otra. Por ejemplo, la clase "Persona" podría estar definida como subclase de la clase "Mamífero". De esto podemos deducir que si un individuo es una Persona, entonces, también es un Mamífero (primer ejemplo de posible inferencia de conocimiento con OWL).
rdf:Property	Las propiedades pueden utilizarse para establecer relaciones entre individuos o de individuos y valores de datos. Ejemplos de propiedades podrían ser "tieneHijo", "tieneFamiliar", "tieneHermano", y "tieneEdad". Los tres primeros pueden utilizarse para relacionar una instancia de la clase Persona con otra instancia de la clase Persona (siendo casos de "ObjectProperty"), y el último (tieneEdad) puede ser usado para relacionar una instancia de la clase "Persona" con una instancia del tipo de datos "Entero" (siendo un caso de "DatatypeProperty"). Ambas, "owl:ObjectProperty" y "owl:DatatypeProperty", son subclases de la clase de RDF "rdf:Property".
rdfs:subPropertyOf	Las jerarquías de propiedades pueden crearse haciendo una o más indicaciones de que una propiedad es, a su vez, sub-propiedad de una o más propiedades. Por ejemplo, "tieneHermano" puede ser una sub-propiedad de "tieneFamiliar". De esta forma, un razonador puede deducir que si un individuo está relacionado con otro por la propiedad "tieneHermano", entonces está también relacionado con este otro individuo por la propiedad "tieneFamiliar".
rdfs:domain	Un dominio de propiedad reduce los individuos a los que puede aplicarse la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a esa clase.

	Por ejemplo, puede establecerse que la propiedad “tieneHijo” tenga como dominio la clase “Mamifero”. De esto, un razonador puede deducir que si "Frank” tieneHijo “Anna", entonces Frank debe ser un Mamífero. Obsérvese que “rdfs:domain” se denomina restricción global debido a que la restricción se refiere a la propiedad y no sólo a la propiedad cuando está asociada con una clase en concreto.
rdfs:range	El rango de una propiedad reduce los individuos que una propiedad puede tener como su valor. Si una propiedad relaciona a un individuo con otro individuo, y está como rango a una clase, entonces el otro individuo debe pertenecer a dicha clase. Por ejemplo, la propiedad “tieneHija” debe establecerse que tiene como rango la clase “Mamífero”. A partir de aquí, un razonador puede deducir que si “Louise” está relacionada con “Deborah” mediante la propiedad “tieneHija”, (por ejemplo, “Deborah” es hija de “Louise”), entonces Deborah es un Mamífero. El rango es, al igual que el dominio, una restricción global.
Individual	Los individuos son instancias de clases, y las propiedades pueden ser usadas para relacionar un individuo con otro. Por ejemplo, un individuo llamado “Deborah” puede ser descrito como una instancia de la clase “Persona” y la propiedad “tieneEmpleador” puede ser usada para relacionar el individuo “Deborah” con el individuo “UniversidadDeStanford”.

Tabla XVII: OWL, características del esquema RDF.

Igualdad / desigualdad

Las siguientes características de OWL Lite están relacionadas con los valores de igualdad y desigualdad permitidos.

Elemento	Significado
equivalentClass	Es posible definir dos clases como equivalentes. Las clases equivalentes tienen las mismas instancias. El valor de igualdad puede ser utilizado para crear clases sinónimas. Por ejemplo, “Coche” puede definirse como equivalentClass (clase equivalente) de “Automóvil”. De esta manera, un razonador puede deducir que cualquier individuo que sea instancia de “Coche”, es también una instancia de “Automóvil” y viceversa.
equivalentProperty	Es posible definir dos propiedades como equivalentes. Las propiedades equivalentes relacionan a un individuo con el conjunto de otros individuos similares. Por ejemplo, “tieneLider” debe definirse como equivalentProperty (propiedad equivalente) de “tienePresidente”. De este modo, un razonador puede deducir que si “X” está relacionado con “Y” por la propiedad “tieneLíder”, “X” está también relacionado con “Y” por la propiedad “tienePresidente” y viceversa. Un razonador también puede deducir que “tieneLíder” es una sub-propiedad de “tienePresidente”, y “tienePresidente” es un sub-propiedad de “tieneLíder”.
sameAs	Es posible definir dos individuos como iguales. Estas construcciones pueden utilizarse para crear un número de nombres diferentes que se refieren al mismo individuo. Por ejemplo, el individuo “Deborah” puede establecerse como un individuo igual a “DeborahMcGuiness”.
differentFrom	Es posible definir un individuo como diferente de otros individuos. Por ejemplo, el individuo “Frank” puede establecerse como distinto de los individuos “Deborah” y “Jim”. Por tanto, si los dos individuos, “Frank” y “Deborah”, constituyen valores para una propiedad que se ha establecido como funcional (de este modo la propiedad tiene como máximo un único valor), entonces hay una contradicción. Establecer explícitamente que los individuos son diferentes entre sí puede ser importante al utilizar lenguajes como OWL (y RDF) que no asumen que los individuos tienen un único nombre exclusivamente. Por ejemplo, sin

	información adicional, un razonador no puede deducir que "Frank" y "Deborah" sean referencias de individuos distintos.
AllDifferent	Se puede definir un número de individuos como mutuamente distintos mediante una indicación "AllDifferent". Por ejemplo, se podría establecer que "Frank", "Deborah" y "Jim" son mutuamente distintos usando la construcción "AllDifferent". Al contrario que la indicación "differentFrom" anterior, esta construcción resaltaría que "Jim" y "Deborah" son distintos (no únicamente que "Frank" es distinto de "Deborah" y que "Frank" es distinto de "Jim"). La construcción "AllDifferent" es particularmente útil cuando hay conjuntos de objetos distintos, y cuando los diseñadores están interesados en reforzar la suposición de nombres únicos dentro de estos conjuntos de objetos. Se usa junto con "distinctMembers" para establecer que todos los miembros de una lista son distintos, y disjuntos en pares. Un ejemplo completo del mundo de la música: <owl:AllDifferent> <owl:distinctMembers rdf:parseType="Collection"> <Opera rdf:about="#Don_Giovanni"/> <Opera rdf:about="#Nozze_di_Figaro"/> <Opera rdf:about="#Cosi_fan_tutte"/> <Opera rdf:about="#Tosca"/> <Opera rdf:about="#Turandot"/> <Opera rdf:about="#Salome"/> </owl:distinctMembers> </owl:AllDifferent>
distinctMembers	Elemento descrito en el punto anterior.

Tabla XVIII: OWL, características de igualdad y desigualdad.

Características de propiedad

Hay identificadores especiales en OWL Lite que se utilizan para proporcionar la información referente a las propiedades y a sus valores. OWL distingue dos clases de categorías de propiedades:

- Propiedades de objeto, que unen individuos con individuos, y
- Propiedades de tipos de datos, que unen individuos con valores de datos.

En la siguiente figura quedan representadas estos dos tipos de propiedades:

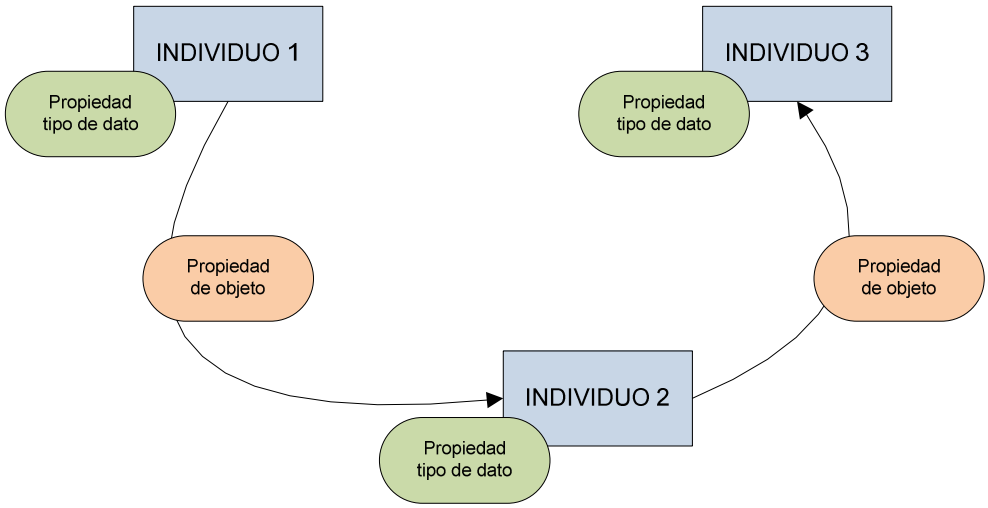


Figura XVI: Propiedad de tipo y de objeto.

Elemento	Significado
ObjectProperty	Propiedad de objeto. Ejemplo: <owl:ObjectProperty rdf:ID="madeFromGrape"> <rdfs:domain rdf:resource="#Wine"/> <rdfs:range rdf:resource="#WineGrape"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="course"> <rdfs:domain rdf:resource="#Meal" /> <rdfs:range rdf:resource="#MealCourse" /> </owl:ObjectProperty>
DatatypeProperty	Propiedad de tipos de datos. Ejemplo: <owl:Class rdf:ID="VintageYear" /> <owl:DatatypeProperty rdf:ID="yearValue"> <rdfs:domain rdf:resource="#VintageYear" /> <rdfs:range rdf:resource="&xsd;positiveInteger"/> </owl:DatatypeProperty> Los siguientes tipos de datos deberían ser utilizados en OWL, por recomendación de la W3C: • xsd:string - Tipo string. • xsd:normalizedString - Tipo string normalizado. • xsd:NMTOKEN - Tipo NM token. • xsd:token - Tipo token. • xsd:Name - Tipo name. • xsd:NCName - Tipo NC name.

	• xsd:integer - Tipo entero. • xsd:int - Tipo entero. • xsd:unsignedInt - Tipo entero sin signo. • xsd:nonPositiveInteger - Tipo entero no positivo. • xsd:nonNegativeInteger - Tipo entero no negativo. • xsd:positiveInteger - Tipo entero positivo. • xsd:negativeInteger - Tipo entero negativo. • xsd:short - Tipo short. • xsd:unsignedShort - Tipo short sin signo. • xsd:long - Tipo long. • xsd:unsignedLong - Tipo long sin signo. • xsd:double - Tipo double. • xsd:float - Tipo float. • xsd:decimal - Tipo decimal. • xsd:hexBinary - Tipo binario hexadecimal. • xsd:base64Binary - Tipo binario en base 64. • xsd:byte - Tipo byte. • xsd:unsignedByte - Tipo byte sin signo. • xsd:dateTime - Tipo fecha y hora. • xsd:gYear - Tipo año. • xsd:gYearMonth - Tipo año y mes. • xsd:gMonth - Tipo mes. • xsd:gMonthDay - Tipo mes y día. • xsd:date - Tipo fecha. • xsd:time - Tipo hora. • xsd:gDay - Tipo día. • xsd:anyURI - Tipo URI. • xsd:Boolean - Tipo booleano. • xsd:language - Tipo lenguaje.
inverseOf	Es posible definir una propiedad como la inversa de otra propiedad. Si se estableciera la propiedad “P1” como inversa de la propiedad “P2”, y relacionáramos “X” con “Y” mediante la propiedad “P2”, entonces “Y” estaría relacionado con “X” mediante la propiedad “P1”, por inferencia. Por ejemplo, si la propiedad “tieneHija” es la propiedad opuesta de “tienePadres” y “Deborah” tienePadres “Louise”, entonces un razonador puede deducir que “Louise” tieneHija “Deborah”. Ejemplo de definición: <owl:ObjectProperty rdf:ID="hasMaker"> <rdf:type rdf:resource="&owl;FunctionalProperty" /> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="producesWine"> <owl:inverseOf rdf:resource="#hasMaker" /> </owl:ObjectProperty>

	En el ejemplo anterior vemos que la propiedad de objeto "producesWine" (produce vino) es una propiedad inversa de la propiedad "hasMaker" (tiene productor).
TransitiveProperty	Es posible definir propiedades como transitivas. Cuando una propiedad es transitiva, si el par (x, y) es una instancia de la propiedad transitiva P, y el par (y, z) es otra instancia de la propiedad transitiva P, entonces el par (x, z) también es una instancia de P. Por ejemplo, si se indica que la propiedad Antepasado es transitiva, si "Sara" es un antepasado de "Louise", es decir, ("Sara", "Louise") es una instancia de la propiedad Antepasado, y si "Louise" es un antepasado de "Deborah", es decir, ("Louise", "Deborah") es una instancia de la propiedad Antepasado, entonces un razonador puede deducir que "Sara" es un antepasado de "Deborah", es decir, ("Sara", "Deborah") es una instancia de la propiedad Antepasado. OWL Lite y OWL DL imponen la condición paralela de que las propiedades transitivas (y sus súper-propiedades) no pueden tener una restricción de cardinalidad máxima (maxCardinality) con valor 1. Sin esta condición paralela, OWL Lite y OWL DL se convertirían en lenguajes no procesables. Ejemplo de definición: <owl:ObjectProperty rdf:ID="locatedIn"> <rdf:type rdf:resource="&owl;TransitiveProperty" /> <rdfs:domain rdf:resource="&owl;Thing" /> <rdfs:range rdf:resource="#Region" /> </owl:ObjectProperty> <Region rdf:ID="SantaCruzMountainsRegion"> <locatedIn rdf:resource="#CaliforniaRegion" /> </Region> <Region rdf:ID="CaliforniaRegion"> <locatedIn rdf:resource="#USRegion" /> </Region> La propiedad de objeto "locatedIn" (localizado en) se define como transitiva para el rango "Region". Por ello, podemos inferir que "SantaCruzMountainsRegion" es una "USRegion".
SymmetricProperty	Es posible definir propiedades como simétricas. Si una propiedad es simétrica, y el par (x, y) es una instancia de esa propiedad simétrica P, entonces el par (y, x) es también una instancia de la propiedad simétrica P. Por ejemplo, "Amigo" puede considerarse una propiedad simétrica. De esta forma, si se indica a un razonador que "Frank" es amigo de "Deborah", éste puede inferir que "Deborah" es amiga de "Frank". Ejemplo de definición: <owl:ObjectProperty rdf:ID="adjacentRegion"> <rdf:type rdf:resource="&owl;SymmetricProperty" /> <rdfs:domain rdf:resource="#Region" /> <rdfs:range rdf:resource="#Region" /> </owl:ObjectProperty> <Region rdf:ID="MendocinoRegion"> <locatedIn rdf:resource="#CaliforniaRegion" /> <adjacentRegion rdf:resource="#SonomaRegion" /> </Region>

	La propiedad de objeto “adjacentRegion” (región adyacente) se ha definido como simétrica. Por ello, podemos inferir que “SonomaRegion” es una región adyacente de “MendocinoRegion”.
FunctionalProperty	Es posible definir propiedades para que tengan un valor único. Si una propiedad es “FunctionalProperty”, ésta no tendrá más de un valor para cada individuo (aunque es posible que no tenga un valor para cada individuo). Esta característica también se denomina propiedad única. FunctionalProperty es una forma abreviada para indicar que la cardinalidad mínima de la propiedad es 0 y la cardinalidad máxima es 1. Por ejemplo, “tieneJefeSuperior” puede establecerse como “FunctionalProperty”. Es, por ello, que un razonador puede deducir que ningún individuo pueda tener más de un jefe principal. Sin embargo, esto no implica que todas las instancias de Persona deban tener al menos un jefe principal. Ejemplo de definición: <owl:Class rdf:ID="VintageYear" /> <owl:ObjectProperty rdf:ID="hasVintageYear"> <rdf:type rdf:resource="&owl;FunctionalProperty" /> <rdfs:domain rdf:resource="#Vintage" /> <rdfs:range rdf:resource="#VintageYear" /> </owl:ObjectProperty> La propiedad “hasVintageYear” (tiene año de vendimia) se ha declarado como funcional. Por ello, de tener, sólo podrá estar asociado a un año de vendimia.
InverseFunctionalProperty	Es posible definir propiedades para que sean funcional inversa. Si una propiedad es funcional inversa, entonces la inversa de la propiedad será funcional. Por tanto, la inversa de la propiedad tiene como máximo un valor para cada individuo. Esta característica también se denomina propiedad inequívoca. Por ejemplo, “tieneNúmerodeSeguridadSocialdeEE.UU” (un identificador único para los residentes en los Estados Unidos) puede establecerse como funcional inversa (o inequívoca). La inversa de esta propiedad (al que se puede llamar “esElNúmeroDeSeguridadSocialPara”) tiene como máximo un valor para cada individuo dentro de la clase de los números de Seguridad Social. De esta manera, el número de Seguridad Social de cualquier persona es el único valor para su propiedad “esElNúmeroDeSeguridadSocialPara”. Es, por esto, que un razonador puede deducir que no existen dos individuos diferentes, instancias de Persona, que tengan idéntico el número de Seguridad Social de los EE.UU. Además, un razonador puede deducir que si dos instancias de Persona tienen el mismo número de Seguridad Social de los EE.UU., es porque dichas instancias se refieren al mismo individuo. Ejemplo de definición: <owl:ObjectProperty rdf:ID="hasMaker" /> <owl:ObjectProperty rdf:ID="producesWine"> <rdf:type rdf:resource="&owl;InverseFunctionalProperty" /> <owl:inverseOf rdf:resource="#hasMaker" /> </owl:ObjectProperty> La propiedad “producesWine” se ha definido como “funcional inversa” de la propiedad “hasMaker”.

Tabla XIX: OWL, características de propiedad.

Restricciones de propiedad

OWL Lite permite establecer restricciones sobre la forma en que las propiedades son utilizadas por las instancias de una clase. Estos elementos (y las restricciones de cardinalidad que veremos en la siguiente sección) se usan dentro del contexto de una restricción con el elemento “owl:Restriction”. El elemento “owl:onProperty” indica la propiedad donde indicaremos las condiciones de restricción. Las dos restricciones indicadas en esta sección indican los valores que pueden ser utilizados, mientras que las restricciones incluidas en la siguiente sección indican la cantidad de valores que pueden ser utilizados.

Elemento	Significado
Restriction	Indica que nos encontramos ante la definición de una restricción. Ejemplo de definición: <owl:Restriction> … </owl:Restriction>
onProperty	Indica el nombre de la propiedad sobre la que vamos a realizar la definición de una restricción. Ejemplo de definición: <owl:Restriction> <owl:onProperty rdf:resource="#hasMaker" /> </owl:Restriction>
allValuesFrom	La restricción “allValuesFrom” se establece sobre una propiedad con respecto a una clase. Esto significa que esta propiedad sobre una determinada clase tiene una restricción de rango local asociada a ella. De este modo, si una instancia de la clase está relacionada con un segundo individuo mediante esa propiedad, entonces puede deducirse que el segundo individuo es una instancia de una clase de la restricción de rango local. Por ejemplo, la clase “Persona” puede tener una propiedad denominado “tieneHija” restringida a tener “allValuesFrom” de la clase “Mujer”. Esto significa que si un individuo “Louise” está relacionado mediante la propiedad “tieneHija” con un individuo “Deborah”, entonces un razonador puede deducir que “Deborah” es una instancia de la clase “Mujer”. Esta restricción permite que la propiedad “tieneHija” sea utilizada por otras clases, como puede ser la clase “Gato”, y tener una restricción de valor apropiada asociada con el uso de la propiedad sobre esa clase. En este caso, “tieneHija” podría tener la restricción de rango local “Gato” cuando se asocie con la clase “Gato” y la restricción de rango local “Persona” cuando se asocie a la clase “Persona”. Obsérvese que un razonador no puede deducir únicamente de la restricción “AllValuesFrom” que realmente existe al menos un valor para la propiedad. Ejemplo de definición: <owl:Restriction> <owl:onProperty rdf:resource="#hasParent" /> <owl:allValuesFrom rdf:resource="#Human" /> </owl:Restriction>
someValuesFrom	La restricción “someValuesFrom” se establece sobre una propiedad con respecto a una clase. Una clase particular puede tener una restricción sobre una propiedad que haga que al menos un valor para esa propiedad sea de un tipo concreto.

	Por ejemplo, la clase "ArticuloSobreWebSemántica" puede tener una restricción "someValuesFrom" sobre la propiedad "tienePalabraClave" que indica que algunos valores de la propiedad "tienePalabraClave" pueden ser instancias de la clase "TemaSobreWebSemántica". Esto permite la opción de tener múltiples palabras claves y, siempre que una o más sean instancias de la clase "TemaSobreWebSemántica", el papel será consistente con la restricción "someValuesFrom". A diferencia de "allValuesFrom", "someValuesFrom" no restringe que todos los valores de una propiedad sean instancias de una misma clase. Si "miArticulo" es una instancia de la clase "ArticuloSobreWebSemántica", entonces "miArticulo" está relacionada por la propiedad "tienePalabraClave" con al menos una instancia de la clase "TemaSobreWebSemántica". Obsérvese que un razonador no puede deducir, como podría hacer con las restricciones "allValuesFrom", que todos los valores de "tienePalabraClave" son instancias de la clase "TemaSobreWebSemántica". Ejemplo: <owl:Restriction> <owl:onProperty rdf:resource="#hasParent" /> <owl:someValuesFrom rdf:resource="#Physician" /> </owl:Restriction>

Tabla XX: OWL, características de restricciones de propiedad.

Cardinalidad restringida

OWL Lite incluye una forma limitada de restricciones de cardinalidad. Las restricciones de cardinalidad de OWL (y OWL Lite) se refieren a restricciones locales, puesto que se definen en las propiedades con respecto a una clase particular. Es decir, las restricciones fuerzan la cardinalidad de esa propiedad sobre instancias de esa clase. Las restricciones de cardinalidad de OWL Lite están limitadas porque permiten solamente realizar indicaciones referentes a cardinalidades de valor "0" o "1" (no está permitido añadir valores arbitrarios para la cardinalidad, como es el caso en OWL DL y OWL Full).

Elemento	Significado
minCardinality (sólo "0" o "1")	La cardinalidad se establece sobre una propiedad con respecto a una clase particular. Si se establece "minCardinality" (cardinalidad mínima) de 1 sobre una propiedad con respecto a una clase, entonces cualquier instancia de esa clase estará relacionada al menos con un individuo mediante esta propiedad. Esta restricción es otra manera de decir que es necesario que la propiedad tenga un valor para todas las instancias de la clase. Por ejemplo, la clase "Persona" no debería tener restricciones de cardinalidad mínima establecidas sobre la propiedad "tieneDescendidente", puesto que no todas las personas tienen descendientes. Sin embargo, la clase "Padre" debería tener una cardinalidad mínima de 1 sobre la propiedad "tieneDescendencia". Si un razonador conoce que "Louise" es una Persona, entonces nada puede ser deducido a partir de la cardinalidad mínima sobre su propiedad "tieneDescendiente". Una vez que se conoce que "Louise" es una instancia de "Padre", un razonador puede deducir que "Louise" está relacionada con al menos un individuo mediante la propiedad "tieneDescendiente". Con sólo esta información, un razonador no puede deducir el número máximo de descendientes para las instancias individuales de la clase "Padre". En OWL Lite las únicas cardinalidades mínimas permitidas son "0" o "1". Una cardinalidad mínima de "0" en una propiedad indica (en ausencia de más información específica) que la propiedad es opcional con respecto a una clase. Por ejemplo, la propiedad "tieneDescendiente" puede tener una cardinalidad mínima de 0 para la clase "Persona" (mientras esté definido tener información más específica con cardinalidad

	mínima de 1 en la clase “Padre”). Ejemplo de definición: <owl:Class rdf:ID="Wine"> <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#madeFromGrape"/> <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality> </owl:Restriction> </rdfs:subClassOf> ... </owl:Class>
maxCardinality (sólo “0” o “1”)	La cardinalidad se establece sobre una propiedad con respecto a una clase particular. Si se establece maxCardinality (cardinalidad máxima) de 1 sobre una propiedad con respecto a una clase, entonces cualquier instancia de esa clase estará relacionada como máximo con un individuo mediante dicha propiedad. Una restricción de maxCardinality 1 es, en ocasiones, denominada una propiedad funcional o propiedad única. Por ejemplo, la propiedad “tieneRegistradoEstadoDeVotación” sobre la clase “CiudadanosdeEstadosUnidos” puede tener una cardinalidad máxima de uno (ya que los ciudadanos sólo pueden votar en un Estado). Por tanto, un razonador puede deducir que instancias individuales de la clase “CiudadanosdeEstadosUnidos” no pueden estar relacionadas con dos o más individuos distintos a través de la propiedad “tieneRegistradoEstadoDeVotación”. Un razonador no puede deducir que la cardinalidad mínima sea 1 solamente de la cardinalidad máxima de 1. Puede ser útil indicar que determinadas clases no tienen valores para una propiedad particular. Por ejemplo, instancias de la clase “PersonaSoltera” no deberían estar relacionadas con ningún individuo mediante la propiedad “tieneCónyuge”. La situación está representada por una cardinalidad máxima de 0 sobre la propiedad “tieneCónyuge” de la clase “PersonaSoltera”. Ejemplo de definición: <owl:Restriction> <owl:onProperty rdf:resource="#hasPadre" /> <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality> </owl:Restriction>
cardinality (sólo “0” o “1”)	La cardinalidad se presenta como una ventaja cuando es útil establecer que una propiedad tiene sobre una clase “minCardinality” 0 y “maxCardinality” 0, o ambos “minCardinality” 1 y “maxCardinality” 1. Por ejemplo, la clase Persona tiene exactamente un valor para la propiedad “tieneMadreBiológica”. De esta manera, un razonador puede deducir que dos instancias distintas de la clase “Madre” no pueden ser valores para la propiedad “tieneMadreBiológica” de la misma persona. Ejemplo de definición: <owl:Class rdf:ID="Vintage"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#hasVintageYear"/> <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf>

	</owl:Class>

Tabla XXI: OWL, características de restricciones de cardinalidad.

Cabeceras e importación de ontologías

En esta sección veremos los elementos relacionados directamente con las cabeceras de las ontologías, dentro del lenguaje OWL.

Elemento	Significado
Ontology	Normalmente, una ontología contiene información descriptiva de la propia ontología. Una ontología en sí es un recurso, y como tal puede ser descrita utilizando propiedades de OWL o de otros namespaces: <owl:Ontology rdf:about=""> <owl:versionInfo> ... </owl:versionInfo> <rdfs:comment>...</rdfs:comment> <owl:imports rdf:resource="..."/> </owl:Ontology> Esta información se la suele denominar "cabecera de la ontología" y se suele colocar al principio de la definición de la ontología. Veamos un ejemplo completo: <owl:Ontology rdf:about=""> <owl:versionInfo>v 1.17 2003/02/26 12:56:51 mdean</owl:versionInfo> <rdfs:comment>An example ontology</rdfs:comment> <owl:imports rdf:resource="http://www.example.org/foo"/> </owl:Ontology> Los elementos incluidos en la definición anterior suelen ser utilizados en las cabeceras de las ontologías.
imports	Esta sentencia permite importar definiciones contenidas en otras ontologías. Cada referencia se corresponde con una URI que especifica el lugar desde donde será importada dicha ontología. Esta sentencia es transitiva en el sentido de que si la ontología "A" importa la ontología "B" y la ontología "B", a su vez, importa la ontología "C", la ontología "A" también importará la ontología "C".

Tabla XXII: OWL, cabeceras e importación de ontologías.

Intersección de clases

OWL Lite dispone de un constructor de intersección; pero, al utilizarlo, limita el uso del lenguaje.

Elemento	Significado
intersectionOf	OWL Lite permite establecer intersecciones entre clases identificadas y entre las restricciones. Por ejemplo, la clase "PersonaEmpleada" se puede describir como la "intersectionOf" (intersección entre) "Persona" y "ObjetosEmpleados" (podrían ser definido como objetos que

	tienen una cardinalidad mínima de 1 en la propiedad "tieneJefe"). A partir de esto, un razonador podría deducir que cualquier "PersonaEmpleada" tiene por lo menos un jefe. Ejemplo de definición: <owl:Class rdf:ID="WhiteWine"> <owl:intersectionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Wine" /> <owl:Restriction> <owl:onProperty rdf:resource="#hasColor" /> <owl:hasValue rdf:resource="#White" /> </owl:Restriction> </owl:intersectionOf> </owl:Class>

Tabla XXIII: OWL, intersección de clases.

Control de versiones

En esta sección veremos los elementos relacionados con el versionado de una ontología.

Elemento	Significado
versionInfo	Sentencia que permite incluir información de la versión actual de la ontología. NOTA: *En el último elemento de esta sección se incluye un ejemplo completo con todos los elementos.*
priorVersion	Sentencia que permite incluir información de la versión anterior de la ontología.
backwardCompatibleWith	Sentencia que indica que esta ontología es compatible con alguna otra versión de dicha ontología (la indicada). Se indica, por lo tanto, una referencia a otra ontología.
incompatibleWith	Sentencia que indica que esta ontología no es compatible con alguna otra versión de dicha ontología (la indicada). Se indica, por lo tanto, una referencia a otra ontología.
DeprecatedClass	La característica "Deprecation" (despreciar sería la traducción literal) se suele utilizar comúnmente en muchos lenguajes de programación, como Java, para indicar que una característica del lenguaje, en nuestro caso una clase o una propiedad, tiene compatibilidad con versiones anteriores de la ontología, pero no debería ser utilizada dado que ya se anuncia que en alguna versión futura ya no aparecerá. De esta forma damos tiempo para que los creadores de los individuos definidos de una ontología concreta puedan realizar la migración correspondiente. "DeprecatedClass" hace referencia a una clase obsoleta, mientras que "DeprecatedProperty" hace referencia a una propiedad obsoleta. Veamos un ejemplo completo real: <owl:Ontology rdf:about=""> <rdfs:comment>Vehicle Ontology, v. 1.1</rdfs:comment> <owl:backwardCompatibleWith rdf:resource="http://www.example.org/vehicle-1.0"/>

	<owl:priorVersion rdf:resource="http://www.example.org/vehicle-1.0"/> </owl:Ontology> <owl:DeprecatedClass rdf:ID="Car"> <rdfs:comment>Automobile is now preferred</rdfs:comment> <owl:equivalentClass rdf:resource="#Automobile"/> <!-- note that equivalentClass only means that the classes have the same extension, so this DOES NOT lead to the entailment that Automobile is of type DeprecatedClass too --> </owl:DeprecatedClass> <owl:Class rdf:ID="Automobile" /> <owl:DeprecatedProperty rdf:ID="hasDriver"> <rdfs:comment>inverse property drives is now preferred</rdfs:comment> <owl:inverseOf rdf:resource="#drives" /> </owl:DeprecatedProperty> <owl:ObjectProperty rdf:ID="drives" />
DeprecatedProperty	Véase el elemento “DeprecatedClass”.

Tabla XXIV: OWL, control de versiones.

Propiedades de anotación

OWL no presenta restricciones a la hora de realizar anotaciones sobre una ontología, permitiéndolo en clases, propiedades, individuos y cabeceras de ontología, pero sólo bajo ciertas circunstancias:

- El conjunto de propiedades de objeto, propiedades de tipos de dato, propiedades de anotaciones y propiedades de ontología deben ser mutuamente disjuntos. Por ejemplo, “dc:creator” no puede ser al mismo tiempo una propiedad de tipo de dato y una propiedad de tipo anotación.
- Las propiedades de anotación deben tener un tipeado explícito, como en el siguiente ejemplo:

  AnnotationPropertyID rdf:type owl:AnnotationProperty

- El objeto asociado a una propiedad de anotación debe ser un literal, una referencia URI, o un individuo.

Las propiedades de anotación permitidas en OWL son las siguientes:

- owl:versionInfo
- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy

Elemento	Significado
rdfs:label	El elemento “label” permite incluir un nombre opcional entendible por un ser humano, en lugar de utilizar el nombre de la clase. Con el atributo “lang” de este elemento podemos incluir

	etiquetas en varios idiomas. Ejemplo de definición: <owl:Class rdf:ID="Wine"> <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/> <rdfs:label xml:lang="en">wine</rdfs:label> <rdfs:label xml:lang="es">vino</rdfs:label> ... </owl:Class>
rdfs:comment	Utilizando este elemento podemos incluir un comentario a una ontología. Ejemplo de definición: <owl:Ontology rdf:about=""> <rdfs:comment>An example OWL ontology</rdfs:comment> <owl:priorVersion rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine"/> <owl:imports rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food"/> <rdfs:label>Wine Ontology</rdfs:label> ...
rdfs:seeAlso	Permite añadir una referencia externa que contiene información adicional sobre cualquier recurso OWL.
rdfs:isDefinedBy	Este elemento es una sub-propiedad del elemento anterior, “seeAlso”, y permite definir el origen o autoridad del elemento anterior.
AnnotationProperty	Elemento que permite incluir en una ontología una propiedad de anotación.
OntologyProperty	Elemento que permite incluir en una ontología una propiedad de la propia ontología.

Tabla XXV: OWL, propiedades de anotación.

1.2.5 Lenguaje HTML5

En esta sección analizaremos una de las partes finales de nuestro programa, la generación de código HTML, en su versión 5. Para empezar debemos responder a la siguiente pregunta:

¿Por qué utilizaremos HTML5 en nuestro generador?

Figura XVII: Logo HTML 5.

HTML 5 (HyperText Markup Language, versión 5) es la quinta revisión del lenguaje de la "World Wide Web" (El desarrollo de este código está regulado por el Consorcio W3C), especificando las siguientes variantes de sintaxis:

- El "clásico" HTML (text/html),
- La variante conocida como HTML5, y
- Una variante XHTML conocida como sintaxis XHTML5 que tiene formato XML (XHTML).

Como iremos comentando a lo largo de esta memoria, la web semántica comienza a ser una realidad. Esta revolución será posible, en parte, gracias a las nuevas características del lenguaje HTML. En su versión 5, las etiquetas semánticas comienzan a tener cierta relevancia. Este lenguaje establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y tradicionales, pero ahora con un significado semántico específico. Veamos algunos ejemplos:

- <header> cabecera de página.
- <nav> bloque de navegación del sitio web.
- <footer> pie de página.

NOTA: *En el apartado de implementación veremos el uso de algunas de estas nuevas etiquetas, las cuales nos permitirán formatear los bloques HTML de la página*.

Novedades

Destaquemos a continuación algunas novedades interesantes del lenguaje:

- Incorpora etiquetas para diseño gráfico (canvas 2D y 3D, audio y video), con codecs para mostrar los contenidos multimedia. De momento, no está determinado el tipo de codecs que finalmente será el estándar, dado que existe una lucha entre imponer codecs gratuitos (WebM + VP8) o de pago (H.264/MPEG-4 AVC).
- Existen nuevas etiquetas para manejar grandes conjuntos de datos: <Datagrid>, <Details>, <menú> y <Command>, las cuales permiten generar tablas dinámicas que pueden ser filtradas, ordenadas y ocultadas desde el lado del cliente (browser).

- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime, etc.) y facilidades para validar el contenido de dichos formularios sin necesidad de utilizar Javascript.
- Nuevos visores, como "MathML" para representar fórmulas matemáticas y "SVG" para incluir en las páginas gráficos vectoriales. En general, se deja abierto el lenguaje para poder incluir e interpretar otros lenguajes XML.
- Posibilidad "Drag&Drop". Nueva funcionalidad para arrastrar objetos como imágenes, conseguida gracias a una nueva API, con gestión mediante el uso de eventos.
- API para trabajar "Off-Line". Permite descargar todos los contenidos necesarios y trabajar en local.
- API para obtener información de Geo-posicionamiento para dispositivos que lo soporten.
- API de almacenamiento (Storage). Posibilidad de almacenamiento persistente en local (equipo con el navegador), con bases de datos (basadas en SQL Lite) o con almacenamiento de objetos por aplicación o por dominio Web (Local Storage y Global Storage). Se dispone, por lo tanto, de una base de datos local con la posibilidad de hacer consultas SQL desde el cliente.
- API para el desarrollo de Sockets web, con comunicación bidireccional entre páginas, similar a los Sockets del lenguaje C.
- Posibilidad de hilos de ejecución en paralelo ("Web Workers").

Y en un futuro próximo está anunciada una nueva API capaz de obtener información del sistema ("System Information API"). Con esta API se podrá acceder al hardware del equipo del navegador a bajo nivel: red, ficheros, CPU, memoria, puertos USB, cámaras, micrófonos entre otros. Muy interesante desde el punto de vista técnico, pero con inconvenientes por razones de seguridad y privacidad.

Una vez decidido que el lenguaje a generar por nuestro programa será HTML5, en los siguientes bloques de sección analizaremos algunas características adicionales. El conocimiento de todos estos elementos HTML serán requeridos a la hora de generar el código HTML automático desde nuestro generador **GPAHS**.

Elementos semánticos

El lenguaje HTML5 incluye nuevas etiquetas para definir contenidos semánticos en las páginas web y, también, para dar formato. Estas etiquetas permiten describir cual es el significado del contenido como, por ejemplo, su importancia, su finalidad y las relaciones que existen. No tienen especial impacto en la visualización, dado que se incluyen únicamente para facilitar la labor de los buscadores. Los buscadores podrán indexar e interpretar esta meta-información para no buscar simplemente apariciones de palabras en el texto de la página, como sucede en la actualidad. Otra característica importante del lenguaje es que permite incorporar a las páginas web contenido RDF/OWL para describir relaciones entre los términos utilizados en la página.

NOTA: *Precisamente esta característica del lenguaje HTML5 fue la que originó la concepción y alcance definido para el PFC.*

Presentamos a continuación las principales etiquetas que nos permiten crear bloques de carácter semántico:

- Etiqueta <header>: Define una región de cabecera de una página o una sección.
- Etiqueta <footer>: Define una región final de una página o una sección.
- Etiqueta <nav>: Define una región de navegación de una página o una sección.
- Etiqueta <section>: Define una región lógica de una página o de un grupo de contenido.
- Etiqueta <article>: Define un bloque de contenido con el sentido de "artículo", el cual representa un elemento completo de contenido.
- Etiqueta <aside>: Define un contenido secundario o un contenido relacionado con otro.
- Etiqueta <meter>: Permite describir una cantidad dentro de un rango de valores.
- Etiqueta <progress>: Permite indicar el progreso de una acción hasta alcanzar un objetivo concreto. De momento, ningún navegar soporta el uso de esta etiqueta.

Además, el lenguaje ofrece la posibilidad de incluir atributos de nuevos tipos de datos. Sobre cualquier elemento de la página web podemos incluir el patrón "data-<atributo>". Esta información podrá ser recuperada vía JavaScript gracias al método getAttribite().

Elementos canvas

Una de las novedades más interesantes del lenguaje HTML5 son las etiquetas "canvas", gracias a la cuales y mediante programación JavaScript podemos dibujar cualquier tipo de elemento gráfico. Se presentan en el siguiente capítulo de la memoria los elementos canvas que serán utilizados en nuestro desarrollo.

2 ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

Este segundo capítulo de la memoria contiene el análisis, diseño e implementación de nuestro generador, y está compuesto de los siguientes apartados:

Contenido	Apartados	Comentarios
Análisis y diseño	2.1	Análisis y decisiones de diseño del generador GPAHS.
Implementación	2.2	Implementación del generador GPAHS.

Tabla XXVI: Apartados del capítulo 2.

2.1 Análisis y diseño

En este apartado veremos todos los elementos que han requerido un análisis y diseño detallado, previo al desarrollo realizado, para determinar la mejor opción basándonos en nuestros objetivos de proyecto. Los elementos analizados han sido los siguientes:

Contenido	Sección	Comentarios
Diseño del formato de salida HTML	2.1.1	En esta primera sección se incluyen las principales características y novedades de este lenguaje, en su versión 5.
Diseño de los analizadores léxico/sintáctico	2.1.2	En esta sección se presentarán las principales decisiones de diseño del analizador léxico y sintáctico.
Diseño tratamiento etiqueta FEN	2.1.3	En esta sección se analiza las distintas vías de actuación relacionadas con el tratamiento del contenido de las etiquetas FEN.
Ontología ChessPosition	2.1.4	Siguiendo un proceso “paso a paso” se define en esta sección la ontología “ChessPosition”, uno de los principales objetivos de este PFC.
Diseño con Protégé	2.1.5	Una vez definida la ontología, mediante el editor Protégé creamos dicha ontología sobre un fichero OWL.
Hoja de estilos CSS	2.1.6	Incluimos en esta sección una breve referencia de la conveniencia de uso de hojas de estilo.
Análisis de programas de representación de partidas de ajedrez	2.1.7	En esta última sección revisamos la apariencia de los programas de uso más genérico dentro del mundo de ajedrez, para visualizar el contenido de ficheros PGN.

Tabla XXVII: Secciones del apartado 2.1.

2.1.1 Diseño formato de salida HTML

Ha llegado el momento de presentar el diseño inicial de nuestra página web. En la siguiente figura presentamos el primer boceto de lo que deseamos obtener. Nuestro diseño contiene los siguientes elementos:

- Una sección inicial con la cabecera de la página, compuesta de un logo, el acrónimo del aplicativo y el nombre del aplicativo. Esta sección podría estar definida con la etiqueta <header> y para la colocación de los elementos en la pantalla se podría utilizar la etiqueta <table>.

- Una sección intermedia con todas las posiciones de partida existentes en el fichero PGN. Como contenedor externo utilizaremos un bloque . Para cada partida, a su vez, se utilizarán nuevos bloques incluidos en el primero. Deseamos igualmente incluir un link en cada una de las posiciones para poder consultar la solución del ejercicio.

- Una sección final con el pie de página, cuyo contenido podría ser el texto "© 2011 UOC".

La apariencia de la página deberá ser algo parecido a lo mostrado en la siguiente figura:

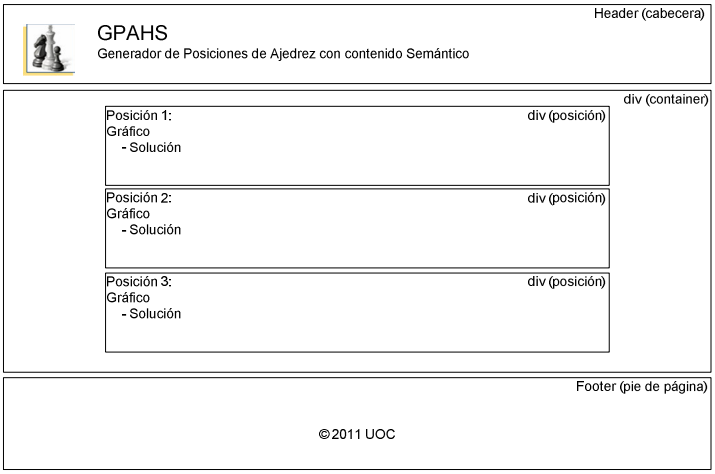


Figura XVIII: diseño inicial página HTML.

NOTA: *Los marcos que aparecen en la figura anterior no son significativos, sólo se han utilizado para facilitar la comprensión del diseño.*

El código HTML5 que permite obtener esta página web, a modo de diseño inicial, es el siguiente:

Código fuente HTML5 (versión 1):

```
<!DOCTYPE html>
<html lang="es">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>GPAHS</title>
 </head>
 <body>
 <header id="page_header">
   <table>
     <tr><td><img src="img/chess.gif" alt="Logo GPAHS" /></td>
       <td><h1>GPAHS</h1></td>
     </tr>
     <tr><td></td>
       <td><h3>Generador de Posiciones de Ajedrez en HTML5 Semántico</h3></td>
     </tr>
   </table>
 </header>
 <div id="container">
  <div class="posicion">
    <p class="posicion_nombre">Posición didáctica 1</p>
    <p class="posicion_gráfico">gráfico de posición</p>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
  </div>
  <div class="posicion">
    <p class="posicion_nombre">Posición didáctica 2</p>
    <p class="posicion_gráfico">gráfico de posición</p>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
  </div>
  <div class="posicion">
    <p class="posicion_nombre">Posición didáctica 3</p>
    <p class="posicion_gráfico">gráfico de posición</p>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
  </div>
 </div>
 <footer id="page_footer">
   <p>&copy; 2011 UOC</p>
 </footer>
 </body>
</html>
```

Utilizando el navegador Safari obtenemos la siguiente página web:

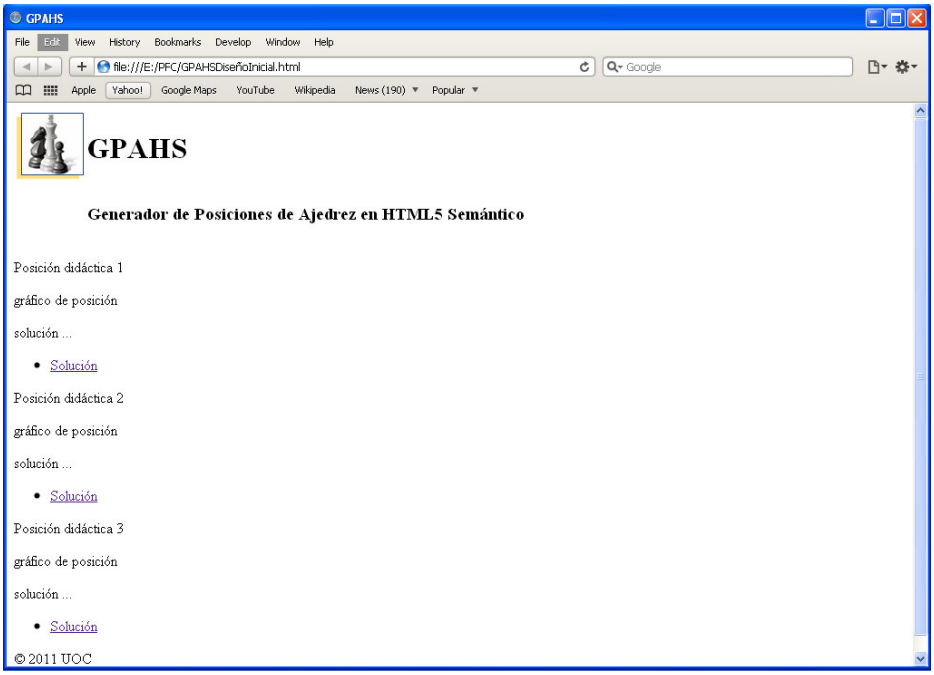


Figura XIX: diseño de página HTML inicial.

Elementos dinámicos del lenguaje HTML5

En el código fuente anterior hemos visto un elemento de navegación. Al pulsar el link "Solución" deseamos mostrar la solución y si el usuario vuelve a pulsar el link ocultaremos su contenido.

Hay varias formas de conseguir este dinamismo, pero casi todas pasan por el uso del lenguaje JavaScript. Nuestro objetivo es la simplificación del código Java requerido para la generación de código HTML, por lo que nos apoyaremos en una utilidad JavaScript con bastante reconocimiento en el mercado: JQuery (versión 1.5.2).

A continuación incluimos la nueva plantilla HTML con las etiquetas <scripts> requeridas para crear el dinamismo comentado anteriormente.

Código fuente HTML (versión 2):

```
<!DOCTYPE html>
<html lang="es">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>GPAHS</title>
  <link rel="stylesheet" href="css/style.css" type="text/css">
  <script type="text/javascript"
        charset="utf-8"
        src="js/jquery-1.5.2.js">
  </script>
  <script type="text/javascript"
        charset="utf-8"
        src="js/GPAHS.js">
  </script>
 </head>
 <body>
```

```
<header id="page_header">
  <table>
    <tr><td><img src="img/chess.gif" alt="Logo GPAHS" /></td>
      <td><h1>GPAHS</h1></td>
    </tr>
    <tr><td></td>
      <td><h3>Generador de Posiciones de Ajedrez en HTML5 Semántico</h3></td>
    </tr>
  </table>
</header>

<div id="container">
 <div class="posicion">
    <p class="posicion_nombre">Posición didáctica 1</p>
    <p class="posicion_gráfico">gráfico de posición</p>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
 </div>
 <div class="posicion">
    <p class="posicion_nombre">Posición didáctica 2</p>
    <p class="posicion_gráfico">gráfico de posición</p>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
 </div>
</div>
<footer id="page_footer">
  <p>&copy; 2011 UOC</p>
</footer>
</body>
</html>
```

La primera etiqueta <script> permite importar la librería "Jquery". La segunda etiqueta <script> permite importar el fichero con las sentencias Jquery que nos permiten crear dinamismo en la página. El contenido de dicho fichero es el siguiente:

Código del fichero GPAHS.js:

```
// GPAHS - Generador de Posiciones de Ajedrez en HTML5 y contenido Semántico
// UOC - 2011
$(document).ready(function(){
 // ocultamos las soluciones de cada posición:
 $(".posicion .posicion_solucion").hide();
$(".actions li.solucion a").click(function(event){
  var $currElement = $("#start");
  $currElement = $(this).parents("ul");
  $currElement = $currElement.prev(".posicion_solucion");
  $currElement.toggle();
  event.preventDefault();
});
});
```

Elementos gráficos del lenguaje HTML5

Gracias a los nuevos elementos canvas del lenguaje HTML5 dibujaremos los dos objetos principales de la posición didáctica: el tablero y las piezas.

Para minimizar el número de sentencias JavaScript requeridas para dibujar el tablero sólo se dibujarán los escaques de color, manteniendo en blanco (color de fondo) el resto. Para dibujar las

piezas utilizaremos imágenes en formato “pgn”. El tamaño de los escaques es de 30x30 píxeles, y el tamaño de las piezas será igualmente de 30x30 píxeles. Los distintos elementos canvas que serán requeridos en nuestro desarrollo son los siguientes:

Declaración de elemento canvas

```
<canvas id="canvas1" width="250" height="250">
        Su browser no acepta elementos canvas del HTML5.
</canvas>
```

Declaración de variable asociada a elemento canvas

```
var ctx = document.getElementById(canvas).getContext('2d');
```

Ejemplo de dibujo de escaques (fila 8 del tablero)

```
ctx.beginPath();
ctx.fillStyle = "rgb(204,204,153)";
ctx.fillRect(31,1,30,30); //fila 8
ctx.fillRect(91,1,30,30);
ctx.fillRect(151,1,30,30);
ctx.fillRect(211,1,30,30);
ctx.fill();
```

Ejemplo de dibujo de imágenes (piezas)

```
var bk = new Image(); // rey negro
...
bk.src = "img/24/bk.png";
...
ctx.drawImage(bk,124,4,24,24);
...
```

NOTA: *En la fase de implementación veremos el desarrollo completo de todos estos* elementos gráficos.

2.1.2 Diseño de los analizadores léxico/sintácticos

Las principales decisiones de diseño relacionadas con el analizador léxico son las siguientes:

- Se debe hacer uso de macros y expresiones regulares que mejoren la legibilidad del código LEX.
- Cada token particular debe ser tratado de forma independiente, excepto en el caso de movimientos de una partida. En este último caso, el conjunto de información contenida en esta sección será tratada como un bloque independiente.

A modo de ejemplo, incluimos a continuación una sección del código fuente del fichero LEX con la definición de algunos tokens que serán utilizados por el analizador sintáctico:

```
…
cadenastring = "\""[^\n"\""]*"\""
digitos = [0-9]+
signo = "+"|"-"
entero = {signo}?{digitos}
flotante = {signo}?{digitos}\.{digitos}
nag = "$"{digitos}
…
[\t \r\n]*  { /* ignorar blancos */ }
"."                 { return new Symbol(sym.TK_PUNTO); }
"*"                 { return new Symbol(sym.TK_ASTERISCO); }
"["                 { return new Symbol(sym.TK_CORCHETE_ABRIR); }
"]"                 { return new Symbol(sym.TK_CORCHETE_CERRAR); }
"("                 { return new Symbol(sym.TK_PARENTESIS_ABRIR); }
")"                 { return new Symbol(sym.TK_PARENTESIS_CERRAR); }
"<"                 { return new Symbol(sym.TK_MENOR); }
">"                 { return new Symbol(sym.TK_MAYOR); }
{nag}               { return new Symbol(sym.TK_NAG,  new String(yytext())); }
{cadenastring}      { return new Symbol(sym.TK_CADENASTRING, new String(yytext().replaceAll("\"", ""))); }
{entero}            { return new Symbol(sym.TK_ENTERO, new String(yytext())); }
{flotante}          { return new Symbol(sym.TK_FLOTANTE, new String(yytext())); }
…
.                   { int line = yyline+1;
                      System.out.println("Error lexico en linea " + line + " - " + "Simbolo desconocido: " + yytext()); }
```

En lo referente al analizador sintáctico, a continuación incluimos la sintaxis formal presentada en la normalización de los ficheros PGN:

```
<PGN-database> ::=  <PGN-game> <PGN-database>
                    <empty>

<PGN-game> ::=              <tag-section> <movetext-section>

<tag-section> ::=           <tag-pair> <tag-section>
                    <empty>

<tag-pair> ::=              [ <tag-name> <tag-value> ]

<tag-name> ::=              <identifier>

<tag-value> ::=             <string>

<movetext-section> ::=      <element-sequence> <game-termination>

<element-sequence> ::=      <element> <element-sequence>
```

```
                          <recursive-variation> <element-sequence>
                          <empty>

<element> ::=             <move-number-indication>
                          <SAN-move>
                          <numeric-annotation-glyph>

<recursive-variation> ::= ( <element-sequence> )

<game-termination> ::=    1-0
                          0-1
                          1/2-1/2
                          *
<empty> ::=
```

NOTA: *Esta sintaxis se ha obtenido de la documentación oficial de definición de los ficheros PGN, por lo que puede ser tomada como un buen punto de partida. El único desarrollo adicional es la transformación de esta sintaxis formal a la sintaxis propia de la utilidad CUP, y el ajuste a las necesidades concretas de nuestro desarrollo*.

2.1.3 Diseño tratamiento etiqueta FEN

La información contenida en el valor asociado a la etiqueta "FEN" nos debe permitir "dibujar" dicha posición sobre una página HTML5, utilizando las nuevas características "canvas" de dicho lenguaje, y gracias también al dinamismo proporcionado por el lenguaje JavaScript incluido en el código HTML5 generado.

NOTA: *El detalle lo veremos en el apartado de implementación.*

El generador, una vez finalizada la fase de análisis (léxico, sintáctico y semántico) tendrá a su disposición la información de cada uno de los campos de una partida, almacenando dicha información temporalmente sobre objetos en memoria. En la fase de síntesis debemos generar el código necesario para representar gráficamente las posiciones indicadas en estos campos (piezas y peones en cada fila, turno de juego, enroques posibles, posición de la captura al paso, medios-movimientos sin captura o sin movimiento de peón, y movimientos (turnos) realizados.

Para obtener dicho gráfico tenemos dos posibles vías de diseño:

1. El generador incluye en la página HTML una función JavaScript (sin argumentos de entrada) que dibujaría la posición y toda la información relacionada con dicha posición. Podríamos decir que esta versión es "estática" en el sentido de que cada función requerida es válida únicamente para una posición concreta, y su contenido es incluido por el generador.
    - Inconveniente: Dado que un fichero PGN puede contener (como ya hemos visto) más de una partida, tendríamos que generar tantas funciones como partidas incluidas.
    - Ventaja: simplificamos el código JavaScript secundario requerido para el generador, dado que simplemente secuenciaríamos un conjunto de etiquetas "canvas".

2. El generador incluye en la página HTML código JavaScript que contiene una función (con argumentos de entrada) que dibujaría la posición y toda la información relacionada con dicha posición basándose en la información recibida como argumento (el string FEN).
    - Ventaja: Un fichero PGN puede contener más de una partida, pero sólo sería necesaria una única función, pero con argumentos de invocación diferentes.
    - Inconveniente: Requerimos desarrollar una función JavaScript más sofisticada.

Entendemos que la segunda opción es mucho más interesante por independizar el proceso de generación de código HTML con la representación gráfica de las posiciones. La función JavaScript puede estar ya definida en un fichero "js" o directamente en el código HTML que será utilizada como plantilla. Lo único que debería realizar nuestro generador es la inclusión del código HTML asociado a la invocación a dicha función. Resaltemos también de esta segunda opción la posibilidad de modificar la forma de representar gráficamente una posición, independientemente de la generación de código HTML desde nuestro generador. La información que será proporcionada a la función JavaScript será precisamente el string FEN, recuperado del fichero PGN.

2.1.4 Ontología ChessPosition

Dado que nos encontramos con una tecnología incipiente, no existe ni una sola forma ni una sola metodología estándar para desarrollar ontologías. En este PFC abordaremos los puntos generales que deben ser tomados en consideración, ofreciendo uno de los procedimientos posibles más utilizados a la hora de desarrollar una ontología concreta, y recomendado por la W3C. Describiremos un enfoque iterativo en el desarrollo de la ontología, comenzando por abordar la ontología de manera frontal, es decir, en su conjunto. A continuación, volveremos sobre dicha ontología y la iremos refinando en un proceso evolutivo, completándola con nuevos detalles cada vez que proceda. A lo largo de este proceso discutimos las decisiones de modelización que toma el diseñador, así como las ventajas, inconvenientes e implicaciones de cada una de las soluciones planteadas.

Inicialmente, queremos enfatizar algunas reglas fundamentales en el diseño de ontologías a las cuales nos referiremos en ocasiones, y que pueden ayudar a la hora de tomar decisiones en muchos casos:

1. Como hemos indicado, no hay una forma correcta y única de modelar un dominio, dado que siempre hay alternativas viables. La mejor solución casi siempre depende de la aplicación que se tiene en mente y de las posibles extensiones que puedan surgir.
2. El desarrollo de ontologías es un proceso necesariamente iterativo.
3. Los conceptos en la ontología deben ser cercanos a los objetos (físicos o lógicos) y relaciones en un dominio de interés. Estos conceptos son probablemente sustantivos (en el caso de objetos) o verbos (en el caso de las relaciones) en oraciones que describen un dominio.

Es decir, determinar para qué vamos a usar la ontología y cómo de detallada o general será la ontología nos guiará en muchas de las decisiones de diseño. Entre las alternativas viables que siempre tendremos, necesitaremos determinar cuál funcionará mejor para la tarea proyectada, cuál será más intuitiva, más extensible y cuál ofrecerá un mejor mantenimiento. Necesitamos también precisar que una ontología es un modelo de la realidad del mundo y los conceptos en la ontología deben reflejar dicha realidad. Después de que hayamos definido una versión inicial de la ontología, podemos evaluarla y depurarla usándola en aplicaciones o métodos que resuelvan problemas o discutiéndola con expertos en el área.

Para este PFC, el proceso de definición de la ontología ha sido discutido con profesionales del mundo del ajedrez, tanto en sus facetas de jugador (con títulos nacionales en su palmarés) como por su experiencia en el ámbito de la enseñanza. Los profesionales consultados han sido los siguientes:

David Martínez Martín: Maestro Internacional de ajedrez (MI)

- ELO actual: 2398
- http://ratings.fide.com/card.phtml?event=2209748
- Profesor de la Escuela de Tecnificación de la Federación Madrileña de Madrid, y colaborador habitual de Chessbase.

Anna Rudolf: Woman Grand Master (WGM).

- ELO actual: 2347.
- http://ratings.fide.com/card.phtml?event=722855
- Campeona absoluta de Hungría, año 2009.

Guillermo Cabeza

- Organizador del circuito anual de ajedrez "Aula Joven".
- http://ratings.fide.com/card.phtml?event=22262989
- Monitor y profesor experimentado de ajedrez.

A continuación presentamos los pasos realizados para definir la ontología de nuestro dominio.

Paso 1: Determinar el domino y alcance de la ontología

El primer paso a la hora de definir una ontología es la definición de su dominio y alcance. Es decir, responder a las siguientes preguntas básicas:

¿Cuál es el dominio que la ontología cubrirá?

¿Para qué usaremos la ontología?

¿Para qué tipos de preguntas la información de la ontología deberá proveer respuestas?

¿Quién usará y mantendrá la ontología?

Las respuestas a esas preguntas pueden cambiar durante el proceso del diseño de la ontología, pero las respuestas iniciales a estas preguntan ayudarán a limitar el alcance del modelo. Consideremos la ontología de "posiciones de ajedrez", objetivo básico de este PFC. El dominio de la ontología es la representación de todas las posiciones de ajedrez con interés didáctico, dado que dichas posiciones permiten representar una situación de ganancia de material o mate en un número determinado de movimientos, sin que sea posible una réplica válida. Estas posiciones permiten mejorar las aptitudes tácticas y estratégicas de los jugadores de ajedrez.

La ontología podrá ser utilizada por cualquier persona que quiera mejorar su fuerza de juego, así como cualquier profesional (monitor, preparador, analizador) que desee instruir a otra persona y requiera material de apoyo.

También es importante mantener los sinónimos de las palabras que serán utilizadas en la ontología, dado que los usuarios que harán uso de ella tendrán un conocimiento del "juego-ciencia" muy distinto. Veamos varios ejemplos:

- Es habitual que los profesionales denominen al peón aislado (por no tener otros peones en celdas contiguas) como peón "isolani". Un profano requerirá acceder a un diccionario para determinar su significado, o hacer uso de esta base de datos de sinónimos.
- El lugar que ocupa una pieza se denomina "celda", "posición" o "escaque" indistintamente.

- Por último, un ejemplo popular. Las personas que se inician en este juego suelen denominar "reina" (por su cercanía a la palabra "rey") a la pieza que en realidad tiene el nombre de "dama" en castellano.

Preguntas de competencia.

Una de las formas de determinar el alcance de la ontología, como hemos comentado, es bosquejando una lista de preguntas que la base de conocimientos basada en la ontología debería ser capaz de responder. Esas preguntas servirán después como prueba de control de calidad:

¿La ontología contiene suficiente información para responder esos tipos de preguntas?

¿Las respuestas requieren un nivel particular de detalle o representación de un área particular?

Las preguntas de competencia son solamente un bosquejo y no necesitan ser exhaustivas. En el dominio de las posiciones de ajedrez están son las posibles preguntas para las que deseamos definir la ontología:

- Muestra todas las posiciones con posibilidad de mate en un solo movimiento.
- Muestra celadas en apertura.
- ¿Hay posiciones del jugador "Sahuquillo, Alejandro" en la base de conocimiento?
- Muestra posiciones de la táctica "Ataque doble".
- Quiero consultar posiciones de tablas en finales de torre.
- ¿Qué tácticas están presentes en la base de conocimiento?
- Muestra un inventario de todas las posiciones de partidas jugadas por "Polgar" con negras.
- Quiero ver las posiciones del evento "FIDE World Championship".
- ¿Hay posiciones partidas jugadas en los Estados Unidos?
- Muéstrame posiciones de partidas jugadas en el año 2010.
- ¿Hay posiciones tácticas presentes en una ronda inicial de un torneo?
- Muestra las posiciones ganadoras de blancas.
- Quiero ver posiciones ganadoras de partidas perdidas finalmente.
- Posición ganadora con un Maestro Internacional en juego.
- Analizar posiciones en las que no exista posibilidad de enroque en ambos bandos.
- Posiciones con jugadores con ELO FIDE superior a 2300.
- Quiero ver el correo electrónico de todos los jugadores de la base de conocimiento.
- Muéstrame posiciones de partidas con un jugador no humano (programa).
- Quiero analizar posiciones de partidas celebradas el tercer día de competición.
- ¿La Universitat Oberta de Catalunya ha sido sponsor de alguna de las partidas contenidas en la base de conocimiento?
- Posiciones ganadoras en torneos del tipo "Open".
- Posiciones ganadoras en finales de torneos.
- Quiero analizar posiciones de partidas jugadas en el tablero "1".
- Posiciones de partidas de la apertura "Gambito de dama".
- Posiciones de partidas de la variante "Gambito de dama aceptado".
- Posiciones de partidas de la subvariante "defensa Petrov".
- Posiciones de partidas con ECO "B01".

- Posiciones de partidas con NIC “SI 3.5”.
- Posiciones que hayan comenzado más tarde de las 19:00 horas.
- Partidas que se hayan celebrado antes del 31/12/2010 (fecha UTC).
- Partidas que se hayan realizado entre las 13:00 y 14:00 horas (hora UTC).
- Posiciones en partidas realizadas con un control de tiempos de 5 minutos, o menos.
- Quiero analizar posiciones de partidas que hayan terminado por abandono.
- Posiciones comentadas (anotadas) por Anna Rudolf.
- Partidas desarrolladas sobre un tablero de ajedrez.
- Posiciones ganadoras en partidas de más de 70 movimientos.

También serían válidas preguntas que combinen las anteriores con conectores “AND” y “OR”.

Como era de esperar, todas estas preguntas están muy relacionadas con la información que hemos presentado en el formato de los ficheros PGN, fuente única de información para definir las clases y crear las instancias de nuestra base de conocimiento.

Paso 2. Considerar la reutilización de ontologías existentes

Casi siempre vale la pena considerar lo que otra persona ha hecho y verificar si podemos refinar, extender o reducir los recursos existentes para nuestro dominio particular a definir. Reutilizar ontologías existentes puede ser un requerimiento si nuestro sistema necesita interactuar con otras aplicaciones que ya utilizan ontologías particulares o vocabularios controlados. Muchas ontologías ya están disponibles en formato electrónico y pueden ser importadas dentro de un entorno de desarrollo de ontologías que se esté utilizando. El formalismo en el cual una ontología está escrita, a menudo, nos interesa puesto que muchos sistemas de representación de conocimiento pueden importar y exportar ontologías. Aun si el sistema de representación de conocimiento no puede funcionar directamente con un formalismo particular, la tarea de traducir una ontología a partir de un formalismo concreto a otro distinto no debería ser una tarea complicada.

Empiezan a surgir bibliotecas de ontologías reutilizables en la Web. Por ejemplo, podemos usar las siguientes bibliotecas ontológicas:

- Ontolingua:
  - http://www.ksl.stanford.edu/software/ontolingua/
- DAML:
  - http://www.daml.org/ontologies/
- UNSPSC:
  - www.unspsc.org
- RosettaNet:
  - www.rosettanet.org
- DMOZ:
  - www.dmoz.org

La ontología que deseamos definir es algo particular y, de momento, de poco interés colectivo. No ha sido posible encontrar ninguna referencia en estas webs que pueda ser utilizada. No obstante, sí se pueden encontrar definiciones de ontologías que se podrían reutilizar parcialmente. Por ejemplo, un

jugador de ajedrez es una persona, y sí disponemos de ontologías que definen las características principales del dominio "persona", una de las clases que podría ser definida en nuestra ontología.

Paso 3. Enumerar los términos importantes para la ontología

Es útil escribir una lista con todos los términos con los que quisiéramos hacer enunciados o dar respuesta a las preguntas de un usuario.

¿Cuáles son los términos de los que quisiéramos hablar?

¿Qué propiedades tienen esos términos?

Por ejemplo, los términos más importantes de nuestra ontología deberían incluir el lugar del encuentro, los jugadores que están celebrando la partida y la posición concreta que deseamos analizar, dentro de dicha partida.

Inicialmente, es importante obtener una lista integral de todos los términos, sin preocuparse del recubrimiento entre los conceptos que representan, relaciones entre los términos, o cualquier propiedad que los conceptos puedan tener, o si los conceptos son clases o slots.

Realicemos esta enumeración de conceptos principales a partir de la información PGN que ya sabemos que vamos a tratar, más información adicional que vamos a requerir en nuestra ontología:

Concepto	Descripción
Event	Nombre del evento donde se realiza el torneo y, por ello, la partida.
Site	Lugar donde se realiza el evento y, por ello, la partida.
Date	Fecha del inicio de la partida.
Round	Ronda, dentro de un evento, en el que se ha realizado la partida.
White	Jugador que juega una partida con piezas blancas.
Black	Jugador que juega una partida con piezas negras.
Result	Resultado de la partida. Victoria para blancas, para negras o tablas.
WhiteTitle	Título asociado al jugador que juega con piezas blancas.
BlackTitle	Título asociado al jugador que juega con piezas blancas.
WhiteELO	ELO asociado con el jugador que juega con blancas.
BlackELO	ELO asociado con el jugador que juega con negras.
WhiteUSCF	ELO de la USCF asociado con el jugador que juega con blancas.
BlackUSCF	ELO de la USCF asociado con el jugador que juega con negras.
WhiteNA	Correo electrónico asociado con el jugador que juega con blancas.
BlackNA	Correo electrónico asociado con el jugador que juega con blancas.
WhiteType	Tipo de jugador (humano o programa) asociado con el jugador que juega con blancas.
BlackType	Tipo de jugador (humano o programa) asociado con el jugador que juega con negras.
EventDate	Fecha de comienzo del evento.

EventType	Tipo de evento.
EventCountry	País donde se desarrolla el evento.
EventRounds	Rondas en las que está organizado el evento.
EventSponsor	Sponsor del evento.
Section	Categoría del evento: Open, Reserve, etc.
Stage	Momento del evento: fase preliminar, semifinales, etc.
Board	Número del tablero, para eventos por equipos o en exhibiciones.
Opening	Nombre de la apertura realizada en la partida.
Variation	Nombre de la variante de apertura realizada en la partida.
SubVariation	Nombre de la sub-variante de la apertura realizada.
ECO	Código de apertura, según la Encyclopedia of Chess Openings.
NIC	Código de apertura, según la New in Chess database.
Time	Hora local de comienzo de la partida.
UTCTime	Hora UTC (Universal Coordinated Time) de comienzo de la partida.
UTCDate	Fecha UTC (Universal Coordinated Time) de comienzo de la partida.
TimeControl	Control de tiempos utilizado en la partida.
Setup	¿Posición de inicio de partida irregular?
FEN	Posición inicial de piezas.
Move	Movimientos restantes a partir de una posición.
Termination	Modo por el que finaliza una partida.
Annotator	Nombre de la persona que ha incluido comentarios en una partida.
Mode	Modo en el que se ha realizado la partida: sobre el tablero, etc.
PlyCount	Número de movimientos realizados en la partida.
Tactic	Táctica relacionada con el tipo de posición presentada: mate en 1, ganancia de material por clavada, etc.
Type	Tipo de posición: ganancia de material, mate en dos movimientos, etc.

Tabla XXVIII: Términos de la ontología ChessPosition.

Paso 4. Definir las clases y la jerarquía de clases

Los siguientes pasos (definición de clases, jerarquías entre ellas y, posteriormente, propiedades) están estrechamente relacionadas entre sí. Es difícil hacer primero una de ellas y luego hacer el resto. Típicamente, creamos unas cuantas definiciones de los conceptos en la jerarquía y, posteriormente, continuamos describiendo las propiedades de esos conceptos y así sucesivamente. Estos pasos son, además, los más importantes en el proceso de diseño de la ontología.

Hay que tener precaución con este punto, dado que es fácil cometer errores cuando se definen clases y jerarquías de clases, dado que no hay una sola jerarquía de clases correcta para un dominio determinado. La jerarquía depende de los posibles usos de la ontología, el nivel de detalle que es

necesario para la aplicación, preferencias personales y, en ocasiones, requerimientos de compatibilidad con otros modelos.

Por ello, discutiremos antes de empezar, varias recomendaciones para tenerlas en cuenta cuando se desarrolle nuestra jerarquía de clases. Después de haber definido un número considerable de nuevas clases, es útil detenerse y verificar si la jerarquía emergente está acorde con estas recomendaciones.

Asegurarse que la jerarquía de clases es correcta

Relación "es-un"

Una jerarquía de clases representa una relación "es-un" ("is-a"). Una clase "A" es una subclase de "B" si cada instancia de "B" también es una instancia de "A". Otra forma de pensar en la relación taxonómica es viéndola como una relación "tipo-de" ("kind-of"). Una subclase de una clase representa un concepto que es un "tipo de" concepto que la superclase representa. Un error común de modelado es el de incluir una versión singular y plural del mismo concepto en la jerarquía. Cuando revisemos la jerarquía como representación de la relación "kind-of" ("tipo-de"), los errores de modelado se detectan rápidamente. Otra forma de evitar este tipo de error es utilizando siempre la forma singular o plural al nombrar las clases.

Transitividad de las relaciones jerárquicas

Una relación de subclase es transitiva si se cumple lo siguiente: si "B" es una subclase de "A" y "C" es una subclase de "B", entonces "C" es una subclase de "A". Algunas veces se hacen distinciones entre subclases directas y subclases indirectas. Una subclase directa es la subclase "más cercana" de la clase padre. No hay clases entre la clase y sus subclases directas en la jerarquía.

Evolución de una jerarquía de clases

Mantener una jerarquía consistente de clases puede llegar a ser desafiante a media que el dominio evoluciona a lo largo del tiempo, dado que las condiciones que determinaron la definición inicial han podido cambiar. Dadas las características de nuestro trabajo, esta circunstancia se tendrá parcialmente en cuenta a la hora de definir nuestra ontología.

Las clases y sus nombres

Es importante distinguir entre una clase y su nombre. Las clases representan conceptos en el dominio y no las palabras que denotan esos conceptos. El nombre de una clase puede cambiar si elegimos una terminología diferente, pero el término, como tal, representa una realidad objetiva en el mundo. Por ejemplo, podemos crear una clase "Camarón" y luego renombrarla a "Gamba" y la clase seguiría representando el mismo concepto. En términos más prácticos, la siguiente regla siempre debe ser seguida: Los sinónimos para el mismo concepto no representan clases diferentes. Los sinónimos son sólo nombres diferentes para un concepto o término. Por lo tanto, no deberíamos tener una clase llamada "Camarón" y una clase llamada "Gamba". Muchos sistemas admiten la asociación de una lista de sinónimos, traducciones, o nombres de presentación con una clase. Si un sistema no permite estas asociaciones, los sinónimos siempre podrían estar listados en la documentación de la clase.

Evitar ciclos en las clases

Debemos evitar ciclos en la jerarquía de clases. Se dice que hay un "ciclo" en una jerarquía cuando una clase "A" tiene una subclase "B" y al mismo tiempo "B" es una superclase de "A". Crear un ciclo como este en una jerarquía equivale a declarar que las clases "A" y "B" son equivalentes, todas las instancias de "A" son instancias de "B" y todas las instancias de "B" son también instancias de "A". En efecto, puesto que "B" es una subclase de "A", todas las instancias de "B" deben ser instancias de la clase "A". Puesto que "A" es una subclase de "B", todas las instancias de "A" deben también ser instancias de la clase "B".

Análisis de clases hermanas en la jerarquía de clases

En este punto analizaremos el concepto de clases "hermanas" y hablaremos sobre la cantidad aconsejable de dichas clases.

Clases hermanas en una jerarquía de clases

Las clases hermanas en una jerarquía son clases que son subclases directas de la misma clase. Todas las clases hermanas en una jerarquía (excepto para las que están al nivel de la raíz) deben estar al mismo nivel de generalidad. Las clases hermanas deben representar conceptos que caen "en la misma línea", de la misma forma que las secciones de un mismo nivel de un libro están al mismo nivel de generalidad. En ese sentido, los requerimientos para una jerarquía de clases son similares a los requerimientos para una estructuración de un libro.

No obstante, existe una excepción. Los conceptos en la raíz de la jerarquía, los cuales son a menudo representados como subclases directas de alguna clase genérica "Cosa" ("Thing"), representan divisiones principales del dominio y no tienen porque ser conceptos similares.

¿Cuándo mucho es demasiado y cuándo poco es insuficiente?

No hay reglas que digan el número de subclases directas que una clase debería tener. Sin embargo, las ontologías bien estructuradas suelen tener entre dos y una docena de subclases directas. Por lo tanto, consideremos las siguientes reglas:

- Si una clase tiene solamente una subclase directa, puede existir un problema de modelado. ¿está completa la ontología?
- Si hay más de una docena de subclases para una clase dada, entonces, podrían ser necesarias categorías adicionales intermedias.

Sin embargo, si no existen clases naturales para agrupar los conceptos en la larga lista de clases hermanas, no hay la necesidad de crear clases artificiales. Después de todo, la ontología es un reflejo del mundo real y si no existen categorizaciones en el mundo real, entonces la ontología debería reflejar eso.

Herencia múltiple

La mayoría de los sistemas de representación de conocimiento admiten herencia múltiple en la jerarquía de clases, es decir, una clase puede ser subclase de varias clases.

¿Cuándo introducir (o no) una nueva clase?

Una de las más difíciles decisiones de tomar durante el modelado es cuándo introducir una nueva clase o cuándo representar una diferencia a través de distintos valores de propiedades. Es difícil navegar en una jerarquía extremadamente anidada con varias clases complejas o en una jerarquía muy plana que tiene pocas clases con mucha información codificada en los slots. Sin embargo, no es fácil encontrar el balance apropiado. Hay varias reglas de base que ayudan a decidir cuándo introducir nuevas clases en la jerarquía. Las subclases de una clase usualmente:

1. Tienen propiedades adicionales que la superclase no tiene,
2. Distintas restricciones a las definidas en la superclase,
3. Participan en relaciones diferentes que las superclases.

En definitiva, introducimos una nueva clase en la jerarquía sólo cuando hay algo que podemos decir acerca de esta clase que no podemos decir de superclase. En la práctica, cada subclase debe tener nuevos slots añadidos a ésta, o tener nuevos valores definidos para el slot, o sustituir algunas facetas de los slots heredados.

Sin embargo, puede ser útil crear nuevas clases cuando no se introducen nuevas propiedades. Las clases en terminologías jerárquicas no necesitan introducir nuevas propiedades. Por ejemplo, algunas ontologías incluyen grandes jerarquías de referencia con términos comunes usados en el dominio. Por ejemplo, una ontología que está relacionada con un sistema de registro electrónico médico puede incluir una clasificación de varias enfermedades. Esta clasificación puede ser solo eso, una jerarquía de términos sin propiedades o un conjunto de valores de propiedad. En este caso, es mejor organizar los términos en la jerarquía porque permite:

1. Una exploración y navegación más sencilla,
2. Facilitará al médico la elección de un nivel de generalidad concreta del término, que sea más apropiada para la situación.

Otra razón para introducir nuevas clases sin nuevas propiedades es para modelar conceptos entre los cuales los expertos del dominio comúnmente hacen una distinción aún cuando no hayamos decidido modelar la distinción en sí. Puesto que usamos las ontologías para facilitar la comunicación entre los expertos de un dominio y entre ellos mismos y los sistemas basados en conocimiento, deseamos reflejar en la ontología la visión del experto sobre el dominio.

Por último, no deberíamos crear subclases de una clase para cada restricción adicional. Cuando definamos una jerarquía de clases, nuestra meta es la de encontrar un balance entre crear nuevas clases útiles para la organización de clases y crear demasiadas clases.

¿Una nueva clase o un valor de propiedad?

Cuando modelamos un dominio, a menudo necesitamos decidir si modelar una distinción específica como un valor de propiedad o como un conjunto de clases, nuevamente, depende del alcance del dominio y de la tarea en mano. La respuesta normalmente está en el alcance que hemos definido para la ontología.

Si los conceptos con diferentes valores de slot se vuelven restricciones para diferentes slots en otras clases, entonces debemos crear una nueva clase para esta distinción. Caso contrario, representaremos la distinción en un valor de slot. Si la distinción es importante en el dominio y

pensamos en los objetos con diferentes valores para la distinción como diferentes tipos de objetos, entonces deberíamos crear una nueva clase para la distinción.

Considerar potenciales instancias individuales de una clase puede ser también útil a la hora de decidir si se introduce una nueva clase o no. Una clase a la cual una instancia individual pertenece no debería cambiar a menudo. Usualmente, cuando usamos propiedades extrínsecas en lugar de intrínsecas de los conceptos para diferenciarlos entre las clases, las instancias de esas clases tendrán que migrar a menudo de una clase a otra. Normalmente, los números, colores, y localizaciones son valores de slots y no conducen a la creación de nuevas clases.

Veamos un ejemplo que clarifica todos estos conceptos. Consideremos una ontología de la anatomía humana. Cuando representamos las costillas:

- ¿Creamos clases para “1ra costilla izquierda”, “2da costilla izquierda” y así sucesivamente?, o
- ¿Creamos una sola clase “Costilla” con slots para indicar el orden y la posición lateral (izquierda-derecha)?

Si la información de cada costilla que representamos en la ontología es significativamente diferente, entonces deberíamos crear una clase para cada una de las costillas. Es decir, si deseamos representar detalles de la información de adyacencia y localización (la cual es diferente para cada costilla) como también funciones específicas que cada costilla juega y órganos que protege, entonces nos interesa la primera opción. Por el contrario, si estamos modelando la anatomía a un nivel básico de generalidad y todas las costillas son muy similares en nuestras aplicaciones potenciales (se trata de ver, por ejemplo, qué costilla está rota en unos rayos X, sin implicaciones con otras partes del cuerpo), entonces se podría simplificar nuestra jerarquía y tener simplemente la clase “Costilla”, con dos slots: “posición lateral” y “orden”.

¿Una instancia o una clase?

Decidir si un concepto particular es una clase en la ontología o una instancia individual depende de cuáles son las aplicaciones potenciales de la ontología. Decidir dónde las clases terminan y las instancias comienzan, empieza por la decisión de cuál es el nivel más bajo de granularidad en la representación. El nivel de granularidad es a su vez determinado por una aplicación potencial de la ontología. En otras palabras:

¿Cuáles son los ítems más específicos que representaremos en la base de conocimientos?

Volviendo a las preguntas de competencia que hemos identificado en el primer paso, los conceptos más específicos que constituirán respuestas a esas preguntas serán buenos candidatos para ser individuos (instancias) en la base de conocimientos. Las instancias individuales son los conceptos más específicos representados en una base de conocimientos.

Otra regla que puede desplazar algunas instancias individuales al conjunto de clases es si los conceptos conforman una jerarquía natural. En este caso, se deberían representar como clases. Solamente las clases pueden ser dispuestas en una jerarquía (los sistemas de representación de conocimiento no tienen la noción de sub-instancias). Por lo tanto, si existe una jerarquía natural entre los términos, debemos definir estos términos como clases, aunque no tengan ninguna instancia propia).

Hay que destacar, por último, que la herramienta Protégé permite a los usuarios especificar algunas clases como abstractas (abstract), indicando que la clase no puede tener ninguna instancia directa. Como veremos más adelante esta característica podría tener utilidad en nuestro modelo.

Limitación del alcance

El siguiente conjunto de reglas es siempre útil a la hora de determinar que una ontología está completa:

- La ontología no tiene porqué contener toda la información posible de un dominio. No se necesita especializar (o generalizar) más de lo que se necesite en la aplicación que hará uso de la ontología.
- De manera similar, la ontología no debe contener todas las posibles propiedades de las clases, ni las distinciones entre ellas en la jerarquía.

Las últimas reglas también se aplican para establecer relaciones entre conceptos que ya los hemos incluido en la ontología. Un nuevo ejemplo para aclarar conceptos, consideremos una ontología que describe "*experimentos biológicos"*. La ontología probablemente contendrá un concepto de o*rganismos biológicos*. También contendrá el concepto de un *experimentador* que ejecuta un experimento (con su nombre, afiliación, etc.). Es cierto que un experimentador es una persona, y como persona, también es casualmente un organismo biológico. Sin embargo, probablemente no debamos incorporar esta distinción en la ontología, para los propósitos de esta representación, un experimentador no es un organismo biológico y probablemente nunca ejecutemos experimentos en los experimentadores como tal. Si estuviésemos representando todo lo que podamos decir de las clases de una ontología, un experimentador llegaría a ser una subclase de organismo biológico. Sin embargo, no necesitamos incluir este conocimiento para las aplicaciones previsibles. De hecho, la inclusión de este tipo de clasificación adicional para las clases existentes es, en realidad, perjudicial dado que una instancia de un experimentador tendrá slots para peso, edad, especie, y otros datos pertenecientes a un organismo biológico, pero absolutamente irrelevantes en el contexto de la descripción de un experimento. Sin embargo, deberíamos registrar esta decisión de diseño en la documentación, para beneficio de los usuarios que analizarán esta ontología y que no estarán al día de la aplicación que teníamos en mente cuando se definió (¿Por qué no se incluyo, o no, el experimentador como organismo biológico?).

Subclases disjuntas

Muchos sistemas nos permiten especificar explícitamente que varias clases sean disjuntas. Las clases son disjuntas si no pueden tener ninguna instancia en común. La especificación de clases disjuntas permite al sistema validar la ontología de una manera más consistente.

Enfoques de diseño

Además de considerar todos los comentarios anteriores, debemos saber que hay varios posibles enfoques a la hora de desarrollar una jerarquía de clases:

- Un proceso de desarrollo *top-down,* comenzando con la definición de los conceptos más generales en el dominio, con la consiguiente especialización de los conceptos.

- Un proceso de desarrollo *bottom-up* comenzando con la definición de las clases más especificas, las hojas de las jerarquías, con el subsiguiente agrupamiento de estas clases en conceptos más generales.
- Un proceso de desarrollo combinado es el resultado de una combinación de los enfoques *top-down* y *bottom-up*: primero definimos los conceptos más sobresalientes y luego los generalizamos y especializamos apropiadamente. Podríamos comenzar, por ejemplo, con unas posiciones de conceptos intermedios como "Apertura", "MedioJuego" y "Final" para realizar especializaciones como "Atracción", "Desviación", "JugadaIntermedia", o generalizaciones como "PosicionTactica".

Ninguno de estos tres métodos es inherentemente mejor que el resto. El enfoque a tomar depende en gran medida con la visión personal del dominio. Si un desarrollador de ontologías tiene una visión sistemática *top-down* del dominio, entonces le será más fácil usar este tipo de enfoque. Lo mismo sucede con el enfoque *bottom-up.* El enfoque combinado es, a menudo, el más fácil para muchos desarrolladores de ontologías, puesto que los "conceptos medios" tienden a ser conceptos más descriptivos en el dominio. Sea cual sea el enfoque que elijamos, debemos saber:

- Se comienza definiendo las clases,
- Se seleccionan los términos que describen objetos que tienen existencia independiente, en lugar de términos que describen esos objetos,
- Estos términos serán las clases,
- El resto de términos serán slots.

Convenios de nomenclatura

Por último, vamos a analizar algunas recomendaciones y convenios a la hora de nombrar las clases y slots. La definición de convenios sobre los nombres de los conceptos en una ontología y su uso estricto, no solamente permiten que la ontología sea fácil de entender, sino que también ayuda a evitar algunos errores comunes de modelado. Existen muchas alternativas para nombrar los conceptos. A menudo, no hay ninguna razón particular para elegir una u otra alternativa. Sin embargo, necesitamos definir un convenio de nombrado de clases y slots, y adherirnos estrictamente a éste. Las siguientes características de un sistema de representación de conocimiento afectan la elección de convenios de nombrado:

- ¿Tiene el sistema el mismo espacio de nombres para las clases, slots e instancias? Es decir, ¿permite el sistema tener una clase y un slot con el mismo nombre?
- ¿El sistema diferencia entre mayúsculas y minúsculas? Es decir, ¿el sistema trata los nombres de manera diferente si están escritos en mayúsculas o minúsculas?
- ¿Qué delimitadores permite el sistema en los nombres? Es decir, ¿los nombres pueden contener espacios, comas, asteriscos y otros caracteres especiales?

Dado que nos apoyaremos en la herramienta Protégé para realizar el diseño utilizaremos sus características. Protégé mantiene un solo espacio de nombres para todos sus marcos, y diferencia entre mayúsculas y minúsculas. Pero esto no siempre sucede. Por ejemplo, la herramienta CLASSIC no diferencia entre mayúsculas y minúsculas y mantiene diferentes espacios de nombres para las clases, slots e individuos (instancias).

Convenio: mayúsculas / minúsculas

Podemos mejorar en gran medida la legibilidad de una ontología si usamos de manera consistente los convenios sobre mayúsculas y minúsculas para los nombres de conceptos. Por ejemplo, es práctica común el escribir en mayúsculas los nombres de las clases y usar minúsculas en los nombres de los slots (asumiendo que el sistema diferencia entre mayúsculas y minúsculas como en nuestro caso).

Cuando el nombre de un concepto contiene más de una palabra necesitamos delimitar las palabras. Presentamos algunas posibles opciones:

- Usar espacios. Protégé permite espacios en los nombres de conceptos.
- Escribir todas las palabras juntas y poner en mayúsculas cada nueva palabra.
- Usar un subrayado o guión u otro delimitador en el nombre.

Si el sistema de representación permite espacios en los nombres, su uso sería la solución más intuitiva para muchos desarrolladores de ontologías. Sin embargo, es importante considerar otros sistemas con los cuales el sistema podrá interactuar. Si estos otros sistemas no usan espacios es mejor no utilizar esta opción. En nuestro caso, utilizaremos la segunda opción por combinar legibilidad y universalidad.

Convenio: Singular / plural

Un nombre de una clase representa una colección de objetos. Por lo tanto, sería más natural para algunos diseñadores nombrar la clase “Personas” en lugar de “Persona”. Ninguna alternativa es mejor o peor que otra, aunque la forma singular para las clases es usada más a menudo en la práctica. Sin embargo, una vez determinada la elección, debe ser consistente a lo largo de toda la ontología. Algunos sistemas solicitan a sus usuarios declarar por adelantado si ellos usarán singular o plural para los nombres de los conceptos y no permiten desviarse de esa elección. El uso de la misma forma todo el tiempo también previene al diseñador de cometer errores de modelado, como ya se ha comentado. Utilizar, por ejemplo, una clase con el nombre “Personas” y definir una subclase de esta como “Persona” sería un error grave pero detectable.

Convenio: Prefijos / sufijos

Algunas metodologías de bases de conocimiento sugieren usar convenios sobre el uso de prefijos y sufijos en los nombres para distinguir entre clases y slots. Dos prácticas comunes a la hora de nombrar los slots son las siguientes:

- Añadir “tiene-“ como prefijo.
- Añadir el sufijo “-de”.

De esta forma, nuestro slots podrían ser “tiene-nombre” o “tiene-posición” si elegimos el convenio que sugiere la opción del prefijo. Si elegimos usar el convenio de los sufijos podríamos tener nombres como “nombre-de” o “posición-de”. Este enfoque permite que los usuarios vean el término para poder determinar inmediatamente si el término es una clase o slot. Sin embargo, tenemos el inconveniente de que los nombres de los términos podrían llegar a ser ligeramente largos.

Otras consideraciones de nombrado

A continuación presentamos algunas consideraciones finales relacionadas con los convenios de nomenclatura:

- No agregar cadenas tales como “clase”, “propiedad”, “slot” a los nombres de conceptos de forma genérica. Siempre está claro, a partir del contexto, si el concepto es una clase o un slot. Si se usan otros convenios de nombrado para las clases y slots como, por ejemplo, el uso de mayúsculas y minúsculas, este convenio artificial no sería necesario.
- Habitualmente es una buena idea evitar abreviaciones de los nombres de conceptos. Por ejemplo, “PS” por “Posición”.
- Los nombres de las subclases directas de una clase deberían todas incluir o no incluir el nombre de la superclase. Por ejemplo, si estamos creando dos subclases de la “Persona” para representar por un lado a los jugadores y por otra a los anotadores, “PersonaJugador” junto con “PersonaAnotador” sería válido, “Jugador” y “Anotador” sería igualmente válido, pero “PersonaJugador” y “Anotador” no sería válido.
- Se debe utilizar un único idioma. “Persona” y “Jugador” es válido dentro de la misma ontología, pero “Persona” y “Player” no lo sería. En nuestro caso, y por proximidad a las etiquetas PGN, se utilizarán todos los términos en inglés.

Teniendo en cuenta todos estos convenios ya tenemos el punto de partida para definir las clases y slots de la ontología.

Definición de clases y jerarquía de clases

NOTA: *En la foto podemos ver en primer término a Alejandro Sahuquillo (ELO >1700), con 10 años es el actual campeón sub12 de la Comunidad de Madrid y firme candidato a ganar el próximo Campeonato de España sub10 (de 27 de junio al 3 de julio de 2011).*

Acompañan en la foto Adler Pérez, Jorge Alejandro Herreros (mi hijo) y David Pujó (todos con ELO >1300), también con buenas opciones para estar entre los 20 mejores jugadores de España en dicho torneo.

Figura XX: Torneo infantil de ajedrez.

Ha llegado el momento de definir las clases y jerarquías de clases que compondrán nuestra ontología, y para ello nos vamos a apoyar en una imagen de un torneo infantil de ajedrez.

Partiendo de un enfoque *top-down,* observemos un evento deportivo de ajedrez para determinar cuáles serían las clases generales iniciales de nuestra ontología. Observando la figura podemos determinar los siguientes elementos generales:

Figura XXI: Objetos ontológicos iniciales.

Evento: lugar donde se celebra un torneo.

Jugador: Persona que disputa una partida.

Tablero de ajedrez: Objeto donde se disputa una partida.

Partida: Movimientos realizados en una partida.

Posición: Posición determinada de una partida (foto).

Anotador: No es un elemento principal que se pueda observar en la imagen anterior, pero ya hemos visto que existe un atributo que hace referencia a esta persona.

Indirectamente, podemos deducir que tanto los jugadores como los anotadores son personas. Analicemos en detalle cada uno de estos conceptos, candidatos para ser definidos como clases.

Evento: Un evento podría contener toda la información relacionada con el lugar donde se está desarrollando un evento deportivo. La clase propuesta tiene el nombre “Event”.

Persona / Jugador / Anotador: Tanto el jugador (en la mayoría de las ocasiones) como el anotador de una partida son personas. Se podría pensar en una clase que englobase a ambos, pero el único atributo que podemos conocer del anotador es su nombre, por lo que no parece necesaria una super-clase de ambas. Además, un jugador de una partida no siempre es una persona. En ocasiones, las “máquinas” también disputan partidas de ajedrez, incluso existen torneos donde sólo juegan este tipo de dispositivos. Además, el anotador de una partida no parece que sea necesaria su identificación como clase. En su lugar, el nombre del anotador será asociado a una de las clases que veremos a continuación. También es importante resaltar que no es necesario tener clases diferenciadas para distinguir los jugadores con piezas blancas de los jugadores con piezas negras. Todo jugador desarrolla partidas en ambos bandos. La clase propuesta para representar a los jugadores tiene el nombre “Player”.

Tablero: Este objeto hace referencia a toda la información relacionada con el tablero de juego, elemento donde normalmente se disputa una partida. Dado que disponemos de información relacionada con este objeto, podemos pensar en definir una clase que la incluya. La clase propuesta tiene el nombre “Board”.

Partida: Llegamos a uno de los objetos más significativos de nuestra ontología. Representa una partida de ajedrez y de ella podemos decir muchas cosas. La clase propuesta tiene el nombre “Match”.

Posición: Hemos llegado a la clase principal de nuestro dominio. La posición de piezas en el tablero es el objeto principal de nuestra ontología para el dominio que deseamos conceptualizar. La clase propuesta tiene el nombre “Position”.

No obstante, con la definición anterior no tendríamos toda la información necesaria. A la hora de definir los requerimientos de nuestra ontología, hemos visto que necesitamos conocer el momento en el que se produce la posición a analizar, así como el tipo de táctica o estrategia a emplear. Para dar solución a este requerimiento definiremos una jerarquía de clases a partir de “Position”. Para definir la jerarquía entre clases debemos evaluar si una instancia que pertenece a una clase determinada, necesariamente lo será siempre de otra clase. Es decir, Si definimos una clase “A” como superclase de la clase “B”, entonces, cada instancia de “B” lo es también de “A”.

Una partida de ajedrez se suele dividir en tres etapas:

- Apertura: Conjunto de movimientos que se realizan al inicio de una partida. En ocasiones, y para poder limitar esta fase de alguna forma, se indica que la fase termina cuando se realiza un enroque en cualquiera de los bandos.
- Final: Conjunto de movimientos que se realizan cuando el número de piezas sobre el tablero nos permiten "calcular" el mejor movimiento a realizar. Existen bases de datos de finales que contienen precisamente este mejor movimiento en cada posible situación (tablas de Nalimov).
- Medio juego: Conjunto de movimientos entre la apertura y el final. Se determina por exclusión de los movimientos correspondientes al resto de etapas.

Obviamente, el límite entre estas tres etapas no siempre es fácil de determinar, y podría ser distinta en función del maestro de ajedrez al que se le preguntase.

Este será nuestro primer nivel de especialización. Nuestra posición concreta tendrá que estar asociada por herencia a alguna de estas tres clases:

- "PositionBegin": Posición asociada a la etapa de apertura.
- "PositionMiddle": Posición asociada a la etapa de medio juego.
- "PositionEnd": Posición asociada a la etapa final del juego.

A partir de esta clasificación inicial incluiremos las posiciones tácticas y estratégicas típicas en cada una de estas etapas.

NOTA: *En este punto del PFC no vemos necesaria la explicación detallada de cada una de estas posiciones, pero se incluye en la memoria un anexo con dicha explicación*.

Por todo ello, concluimos este punto del diseño de la ontología presentando un cuadro con todas las clases definidas, pero antes algunos comentarios finales:

- En realidad, una táctica incluida en la etapa de medio juego se podría aplicar a la etapa de apertura o a la etapa final, pero se ha definido de esta forma para simplificar la ontología final y por ser la que normalmente se presenta en los libros de texto.
- En esta ontología se utilizan los nombres de las tácticas y estrategias en inglés, siguiendo con las recomendaciones anteriores. En el anexo antes mencionado, se presenta también la traducción del término al castellano.
- En general, cuando una posición determinada no se corresponda con ninguna clase de último nivel, asociaremos la posición a la clase superior, así hasta llegar a la clase raíz "Position". Por ello, no se definirá ninguna de las clases como abstractas.

Tras estos comentarios, presentamos gráficamente todas las clases que finalmente han sido definidas en nuestra ontología:

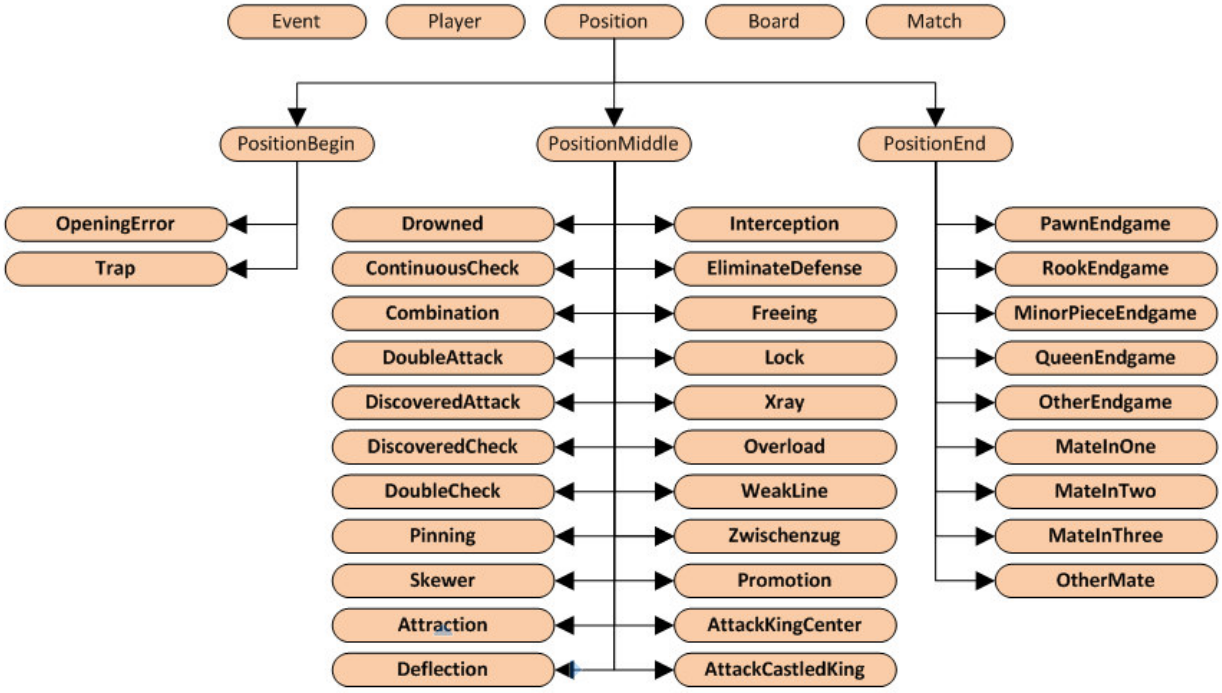


Figura XXII: Jerarquía de clases.

Paso 5. Definir las propiedades de las clases: slots

Las clases aisladas, por sí solas, no proveerán suficiente información para responder las preguntas de competencia presentadas en el paso 1. Una vez que hemos definido las clases, debemos describir la estructura interna de estos conceptos.

Ya hemos seleccionado clases a partir de la lista de términos creada en el paso 3. La mayoría de los términos restantes serán, probablemente, propiedades de estas clases. Para cada propiedad de la lista, debemos determinar qué clase está asociada a la misma.

En general, hay varios tipos de propiedades de objeto que pueden llegar a ser slots en una ontología:

- Propiedades “intrínsecas”.
- Propiedades “extrínsecas”.
- Partes, si el objeto es estructurado; pueden ser “partes” físicas y abstractas.
- relaciones con otros individuos; éstas son las relaciones entre miembros individuales de una clase y otros ítems.

De esta forma, además de las propiedades que hemos identificado previamente, necesitamos añadir nuevos slots para contemplar todas estas propiedades.

Todas las subclases de una clase heredan los slots de esa clase. Un slot deberá estar definido en la clase más general que pueda tener esa propiedad. Pero hay muchos más detalles a tener en cuenta, como veremos a continuación, como es el concepto de "role inverso" y valores por defecto de un slot.

Slots inversos

Un valor de un slot puede depender de un valor de otro slot. Almacenar la información "en ambos sentidos" es redundante, una aplicación que usa una base de conocimientos puede siempre inferir el valor de la relación inversa. Sin embargo, desde la perspectiva de adquisición de conocimiento, es conveniente tener ambas piezas de la información disponibles explícitamente. El sistema de adquisición de conocimiento podría completar automáticamente el valor de la relación inversa asegurando la consistencia de la base de conocimientos. Por ejemplo, un slot "hijo_de" podría tener un slot inverso con el concepto "Padre_de".

Valores por defecto

Muchos sistemas basados en marcos permiten la especificación de valores por defecto para los slots. Si un valor particular de un slot es el mismo para la mayoría de las instancias de una clase, podemos entonces definir un valor por defecto para dicho slot. Entonces, cuando cada nueva instancia de una clase que contenga este slot sea creada, el sistema lo llenará con el valor por defecto automáticamente. Luego se puede cambiar este valor a otro valor en las facetas que lo permitan. Es decir, los valores por defecto están ahí por comodidad, no imponen ninguna nueva restricción sobre el modelo ni cambian el modelo de ninguna forma.

Análisis de slots

A partir de esta información, en las siguientes secciones veremos los slots que han sido definidos para cada una de las clases.

Clase "Event" (Evento)

Un evento puede contener toda la información relacionada con el lugar donde se está desarrollando un evento deportivo. La clase "Event", por lo tanto, contiene la siguiente información:

Slot	Descripción
eventName	Nombre del evento donde se realiza el torneo y, por ello, la partida.
site	Lugar donde se realiza el evento y, por ello, la partida.
eventDate	Fecha de comienzo del evento.
eventType	Tipo de evento.
eventCountry	País donde se desarrolla el evento.
eventRounds	Rondas en las que está organizado el evento.
eventSponsor	Sponsor del evento.

section	Categoría del evento: Open, Reserve, etc.
stage	Momento del evento: fase preliminar, semifinales, etc.

Tabla XXIX: Slots de la clase "Event".

Clase "Player" (Jugador)

La clase "Player" contendrá la siguiente información:

Slot	Descripción
name	Nombre del jugador.
title	Título asociado a un jugador.
elo	ELO asociado a un jugador.
uscf	ELO de la USCF asociado a un jugador.
na	Correo electrónico de un jugador.
typePlayer	Tipo de jugador (humano o programa).

Tabla XXX: Slots de la clase "Player".

Clase "Board" (Tablero)

Esta clase, como vimos, hace referencia a la información relacionada con el tablero de juego, elemento donde normalmente se disputa una partida. La clase "Board" contendrá la siguiente información:

Slot	Descripción
boardNum	Número del tablero, para eventos por equipos o en exhibiciones. NOTA: *Para distinguir el slot del nombre de la clase, se añade "Num" al nombre.*
mode	Modo en el que se ha realizado la partida: sobre el tablero, etc.

Tabla XXXI: Slots de la clase "Board".

Clase "Match" (Partida)

Esta clase representa una partida de ajedrez, y podría contener la siguiente información:

Slot	Descripción
date	Fecha del inicio de la partida.
round	Ronda, dentro de un evento, en el que se ha realizado la partida.
result	Resultado de la partida. Victoria para blancas, para negras o tablas.
opening	Nombre de la apertura realizada en la partida.

variation	Nombre de la variante de apertura realizada en la partida.
subVariation	Nombre de la sub-variante de la apertura realizada.
eco	Código de apertura, según la Encyclopedia of Chess Openings.
nic	Código de apertura, según la New in Chess database.
time	Hora local de comienzo de la partida.
utcTime	Hora UTC (Universal Coordinated Time) de comienzo de la partida.
utcDate	Fecha UTC (Universal Coordinated Time) de comienzo de la partida.
timeControl	Control de tiempos utilizado en la partida.
setup	¿Posición de inicio de partida irregular?
termination	Modo por el que finaliza una partida.
annotator	Nombre de la persona que ha incluido comentarios en una partida.
plyCount	Número de movimientos realizados en la partida.

Tabla XXXII: Slots de la clase "Match".

Clase "Position" (Posición)

Como hemos visto anteriormente, se ha definido una jerarquía de clases para representar una posición de ajedrez. Para que una posición concreta (individuo) pueda estar asociada a cualquier clase de la jerarquía, definimos los slots en la clase raíz "Position", que podrán ser heredados a cualquier subclase. Los slots asociados a esta clase son los siguientes:

Slot	Descripción
Fen	Posición inicial de piezas.
typePosition	Tipo de posición: estratégica (strategy), táctica (tactic) o final (final). También puede ser indeterminada (undetermined) cuando sea una posición genérica.
move	Movimientos restantes a partir de una posición. Representan la solución de la posición en nuestra ontología.

Tabla XXXIII: Slots de la clase "Position".

Destaquemos que el atributo inicial "Position" se ha convertido en una jerarquía de clases, dado que nos interesa guardar información jerárquica y mediante instancias no están permitas las relaciones de este tipo.

Por último, debemos definir los slots de objeto que nos permitirán relacionar todas las clases con nuestra clase principal, "Position". Una posición se desarrolla en una partida concreta, sobre un tablero, con un par de jugadores y en un evento particular. La clase "Position" contendrá los siguientes slots para permitir todas estas relaciones.

Slot	Descripción
hasMatch	Una posición se desarrolla sobre una partida concreta.

hasBoard	Una posición se desarrolla en un tablero concreto.
hasPlayerWhite	Una posición se disputa con un jugador de piezas blancas.
hasPlayerBlack	Una posición se disputa con un jugador de piezas negras.
hasEvent	Una posición se presenta en un evento concreto.

Tabla XXXIV: Slots de objeto de la clase "Position".

Con estas últimas definiciones hemos completado esta fase de la definición de nuestra ontología. En el siguiente paso veremos la definición de facetas de estos slots.

Paso 6. Definir las facetas de los slots

Por último, los slots puedes tener diferentes facetas que describen:

- El tipo de valor,
- Valores admitidos,
- El número de los valores (cardinalidad),
- Otras características de los valores que los slots pueden tomar.

Por ejemplo, el valor del slot "elo" es un número. Es decir, es un slot con "Number" como tipo de valor. Pero antes de realizar la definición de facetas, veamos algunas características de las mismas.

Cardinalidad del slot

La cardinalidad de un slot define cuantos valores puede tener. Algunos sistemas solamente distinguen entre cardinalidad simple (admitiendo sólo un valor) y cardinalidad múltiple (admitiendo cualquier cantidad de valores).

Algunos sistemas admiten la especificación de una cardinalidad mínima y máxima para describir la cantidad de valores de un slot con más precisión. Una cardinalidad mínima "N" significa que un slot debe tener, al menos, "N" valores. Una cardinalidad máxima "M" significa que un slot puede tener, como mucho, "M" valores. Algunas veces puede ser útil fijar la máxima cardinalidad en 0, lo cual significa que el slot no puede tener ningún valor particular para una subclase particular.

Tipo de valor de los slots

Una faceta "tipo de valor" describe qué tipos de valores puede contener un slot. Los tipos de valores más comunes son los siguientes:

- String: es el tipo de valor más simple para los slots, por ejemplo, para nombres. El valor es una simple cadena de caracteres.
- Number: describe slots con valores numéricos. Algunas veces los tipos de valores "Float" e "Integer" se utilizan por ser más específicos.
- Boolean: los slots del tipo "Boolean" son simples banderas "si/no".
- Enumerated: los slots del tipo "Enumerated" especifican una lista específica de valores admitidos para el slot. Debemos destacar que en Protégé, los slots enumerados son del tipo Symbol.

- Instance: los slots del tipo "Instance" admiten la definición de relaciones entre individuos. Deben, por lo tanto, definir una lista de clases admitidas de las cuales las instancias pueden provenir. Algunos sistemas solo especifican el tipo de valor con la clase en lugar de exigir un enunciado especial de slots de tipo instancia.

Dominio y rango de un slot

Las clases admitidas para los slots de tipo "Instance" son a menudo llamadas "rango de un slot". Algunos sistemas permiten restringir el rango de un slot cuando el slot está asociado con una clase particular. Las clases a las cuales un slot está asociado, o las clases cuyas propiedades son descritas por un slot son llamadas "dominio del slot".

Las reglas básicas para determinar un dominio y un rango de un slot son similares. Cuando se define un dominio o rango de un slot, se debe encontrar la clase o clases más generales que puedan ser respectivamente el dominio o rango de los slots. Por otro lado, no debemos definir un dominio o un rango que sea demasiado general. Todas las clases en el dominio de un slot deben ser descritas por el slot y las instancias de todas las clases en el rango de un slot deben poder ser entradas potenciales del slot. No se debe elegir una clase demasiado general para el rango, pero tampoco debemos elegir una clase que no cubra todos los valores posibles de entrada.

En términos más específicos:

- Si una lista de clases que definen un rango o un dominio de un slot incluye una clase y sus subclases, debemos no incluir la subclase.
- Si una lista de clases que definen un rango o dominio de un slot contiene todas las subclases de una clase determinada, el rango debería contener solamente dicha clase (superclase) y no todas sus subclases.
- Si una lista de clases que definen un rango o dominio de un slot contiene unas cuantas subclases de una clase determinada, hay que analizar si dicha clase podrá dar una definición de rango mucho más apropiada.

Definamos a continuación los tipos de datos de todos los slots de nuestro dominio.

Clase "Event"

Slot	Tipo de dato
eventName	String
site	String
eventDate	String
eventType	String
eventCountry	String
eventRounds	String
eventSponsor	String
section	String
stage	String

Tabla XXXV: Tipos de datos de la clase "Event".

Clase "Player"

Slot	Tipo de dato
name	String
title	String
elo	Integer
uscf	Integer
na	String
typePlayer	String

Tabla XXXVI: Tipos de datos de la clase "Player".

Clase "Board"

Slot	Tipo de dato
boardNum	Integer
mode	String

Tabla XXXVII: Tipos de datos de la clase "Board".

Clase "Match"

Slot	Tipo de dato
date	String
round	Integer
result	String
opening	String
variation	String
subVariation	String
eco	String
nic	String
time	String
utcTime	String
utcDate	String
timeControl	String
setup	Integer
termination	String
annotator	String
plyCount	Integer

Tabla XXXVIII: Tipos de datos de la clase "Match".

Clase "Position"

Slot	Descripción
fen	String
move	String
typePosition	String
hasMatch	Instancia de la clase "Match".
hasBoard	Instancia de la clase "Board"
hasPlayerWhite	Instancia de la clase "Player"
hasPlayerBlack	Instancia de la clase "Player"
hasEvent	Instancia de la clase "Event"

Tabla XXXIX: Tipos de datos de la clase "Position".

Paso 7. Crear instancias

El último paso es simple, y consiste en crear instancias individuales de las clases definidas. La definición de una instancia individual de una clase requiere:

1. elegir una clase,
2. crear una instancia individual de la clase, y
3. rellenar los valores del slot.

En el apartado de implementación veremos cómo nuestro generador creará de manera automática todas las instancias de la ontología que hemos definido, a partir de los datos del fichero PGN.

Conclusiones

En esta sección, hemos descrito una metodología de desarrollo de ontologías para sistemas declarativos basados en marcos. Hemos revisado los pasos del proceso de desarrollo de ontologías y hemos descrito los aspectos complejos de la definición de jerarquías de clases, y propiedades de clases, e instancias. Sin embargo, después de seguir todas estas reglas y sugerencias, una de las cosas más importantes a recordar es la siguiente:

No hay una única ontología correcta para un dominio dado.

El diseño de ontologías es un proceso creativo y dos ontologías diseñadas por personas diferentes no serán iguales. Las aplicaciones potenciales de la ontología y el entendimiento y aspectos del dominio desde el punto de vista del diseñador afectarán, sin duda, las opciones de diseño de la ontología.

No se sabe si algo es bueno hasta que se lo pone a prueba.

Podemos evaluar la calidad de nuestra ontología solamente usándola en aplicaciones para las cuales la diseñamos.

Con el análisis de la ontología de ajedrez terminado, podemos proceder a su diseño haciendo uso de la herramienta Protégé.

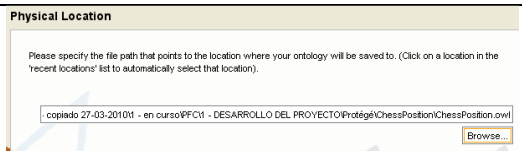
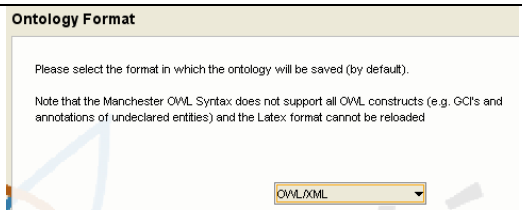
2.1.5 Definición de la ontología con el editor Protégé

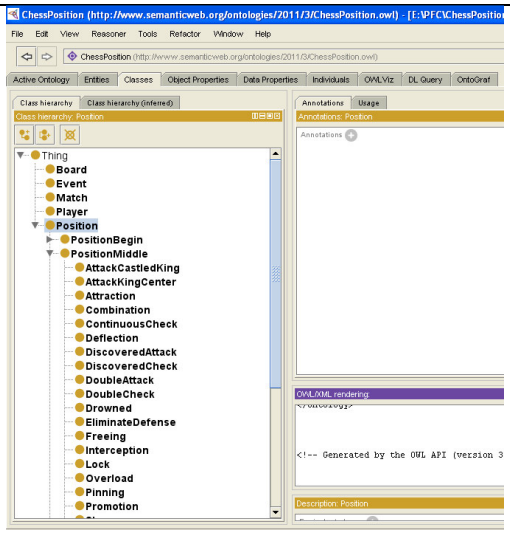
El editor Protégé-OWL es una extensión del propio Protégé que permite la definición de ontologías a través del lenguaje OWL. El editor Protégé-OWL nos permite:

- Cargar y guardar ontologías OWL y RDF.
- Editar y visualizar clases, propiedades y reglas SWRL.
- Definir las características lógicas de clase como expresiones OWL.
- Ejecutar razonadores como clasificadores de lógica de descripción.
- Editar los individuos de marcado OWL.

La arquitectura flexible de Protégé-OWL hace que sea fácil la configuración y ampliación de la herramienta. Protégé-OWL está estrechamente integrado con Jena y tiene un código abierto en Java (API), el cual permite el desarrollo de componentes de interfaz de usuario personalizados, así como el desarrollo de servicios de la Web Semántica.

En esta sección veremos cómo definir todos los elementos analizados en este apartado (clases, slots y facetas). Esta presentación será realizada a modo de tutorial, dado que pensamos que presenta un interés especial dentro del desarrollo de este PFC.

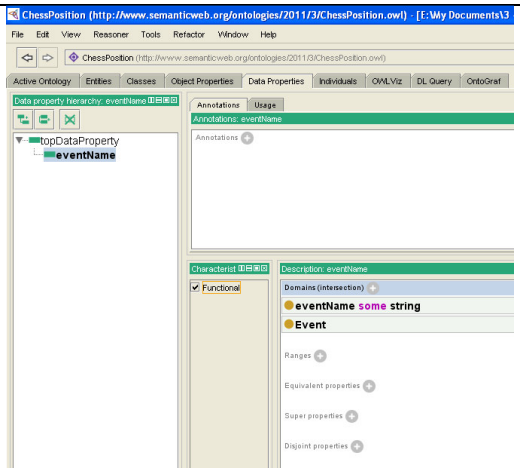
	Cuando arrancamos la herramienta Protégé debemos indicar la actividad que deseamos realizar. En nuestro primer contacto con este editor seleccionaremos la opción de creación de una nueva ontología, pero en accesos posteriores indicaremos la opción de abrir una ontología ya definida previamente.
	El primer paso que debemos realizar es el de la especificación del "Ontology IRI". Este descriptor nos permite identificar una ontología en el contexto de la World Wide Web. Este descriptor debe coincidir con la URL donde definamos la última versión de nuestra ontología publicada. El Ontology IRI propuesto en este proyecto es el siguiente: http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl
	A continuación, indicamos la ubicación física donde será almacenado nuestra ontología, fichero "owl".
	En el siguiente paso indicamos el formato de nuestra ontología. OWL/XML es el que utilizaremos, dado que es la sintaxis que hemos ido utilizando a lo largo de la memoria.



Definición de clases:

Con estas acciones accedemos a la pantalla principal del IDE. Lo primero que vamos a realizar es la definición de las clases.

Pulsamos sobre la pestaña "Classes" para acceder a la edición de clases, y mediante el botón vamos creando a partir de la clase genérica "Thing" y respetando la jerarquía que hemos definido, todas las clases definidas en la sección anterior.

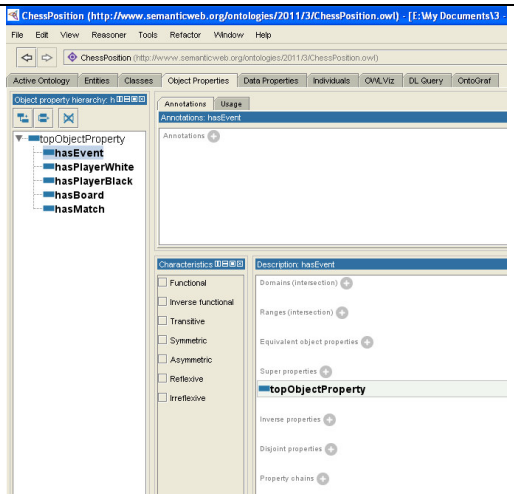


Definición de propiedades de tipos de datos:

Tras la definición de las clases, pulsamos la pestaña "data Properties" para definir los slots de tipos de datos que hemos definido anteriormente para nuestra ontología.

Veamos, a modo de ejemplo como definimos la propiedad "eventName". Tras incluirla en el árbol de propiedades de la sección izquierda de la pantalla, pulsamos el botón de la sección "domains" para incluir dos restricciones. La primera se trata de una restricción de tipo de dato para indicar que la propiedad sólo acepta valores de tipo "String". La segunda restricción es del tipo clase, para asociar esta propiedad exclusivamente a la clase "Event". Por último, indicamos la propiedad como funcional para indicar que si la propiedad tiene valor, sólo aceptará uno.

De este mismo modo procedemos a definir todas las propiedades del dominio, definiendo en todas ellas una clase restrictiva, un tipo de dato asociado, y el carácter de funcionalidad, siempre que proceda.



Definición de propiedades de objetos:

A continuación definimos las propiedades de objetos. En nuestra ontología necesitamos asociar la clase "Position" con el resto de clases, un evento ("Event"), dos jugadores ("Player"), un tablero ("Board") y una partida ("Match").

La definición de este tipo de propiedades se realiza en la pestaña "Object Properties". Una vez creadas todas las propiedades como sub-propiedades de la propiedad genérica "topObjectsProperty", debemos indicar el dominio de aplicación, que en todos los casos será la clase "Position", y el rango de aplicación de cada propiedad. Los rangos son los siguientes:

Propiedad hasEvent	→ rango clase Event.
Propiedad hasPlayerWhite	→ rango clase Player.
Propiedad hasPlayerBlack	→ rango clase Player.
Propiedad hasBoard	→ rango clase Board.
Propiedad hasMatch	→ rango clase Match.

Todas estas propiedades tienen la característica, de nuevo, de ser

	funcionales.
Actividades adicionales: Las facetas o restricciones adicionales de cada uno de los slots, así como comentarios de cada elemento definido, serán incluidos en la fase de desarrollo, una vez iniciemos las pruebas unitarias con los ficheros de entrada previstos.	

Tabla XL: Definición de la ontología con el editor Protégé.

Tras la definición de nuestra ontología mediante la herramienta Protégé, mostramos a continuación los distintas secciones del fichero fuente de dicha ontología, con el renderizado OWL/XML.

El fichero se inicia con la definición de documento XML:

```
<?xml version="1.0"?>
```

A continuación mostramos la cabecera de la ontología:

```
<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl">
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
```

Sección de declaración de clases:

```
   <Declaration>
      <Class IRI="#AttackCastledKing"/>
   </Declaration>
   <Declaration>
      <Class IRI="#AttackKingCenter"/>
   </Declaration>
```

Sección de declaración de propiedades de objeto:

```
   <Declaration>
      <ObjectProperty IRI="#hasBoard"/>
   </Declaration>
   <Declaration>
      <ObjectProperty IRI="#hasEvent"/>
   </Declaration>
```

Sección de declaración de propiedades de datos:

```
   <Declaration>
      <DataProperty IRI="#eco"/>
```

```
</Declaration>
<Declaration>
  <DataProperty IRI="#elo"/>
</Declaration>
```

Sección de declaración de jerarquías de clases:

```
<SubClassOf>
  <Class IRI="#AttackKingCenter"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Attraction"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
```

Sección de declaración de propiedades de objeto funcionales:

```
<FunctionalObjectProperty>
  <ObjectProperty IRI="#hasBoard"/>
</FunctionalObjectProperty>
<FunctionalObjectProperty>
  <ObjectProperty IRI="#hasEvent"/>
</FunctionalObjectProperty>
```

Sección de declaración de dominios de propiedades de objeto:

```
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasBoard"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasBoard"/>
    <Class IRI="#Position"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasEvent"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasBoard"/>
    <Class IRI="#Position"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyDomain>
```

Sección de declaración de rangos de propiedades de objeto:

```
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasBoard"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasBoard"/>
    <Class IRI="#Board"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasEvent"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty IRI="#hasEvent"/>
    <Class IRI="#Event"/>
  </ObjectSomeValuesFrom>
</ObjectPropertyRange>
```

Sección de declaración de propiedades de datos funcionales:

```
<FunctionalDataProperty>
```

```
    <DataProperty IRI="#annotator"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#boardNum"/>
  </FunctionalDataProperty>
```

Y por último, sección de declaración de dominio y tipo de dato de las propiedades de datos:

```
  <DataPropertyDomain>
    <DataProperty IRI="#annotator"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#annotator"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#annotator"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
```

NOTA: *Se incluye como anexo el contenido completo de la definición de nuestra ontología*.

2.1.6 Hoja de estilos CSS

Para minimizar el código HTML a generar, y siguiendo las recomendaciones actuales de desarrollo, todos los estilos de la página estarán incluidos en un fichero "css" externo. A continuación, incluimos un ejemplo de lo que podría ser el formato del elemento HTML que contiene el nombre de cada posición:

Código fuente CSS:

```
p.posicion_nombre{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:20px;
  font-weight:bold;
  color:rgb(204,204,153);
  line-height: 150%;
  text-align:left;
  text-decoration:underline;
}
```

Brevemente, describiremos cada uno de los elementos incluidos en fragmento de código CSS anterior:

- Font-family: Familia fuente de letra. Los valores indicados marcan el orden de selección. Si el primer valor no es reconocible por el browser, se utilizará el segundo valor y así, sucesivamente, hasta encontrar un valor válido. En nuestro caso, "Verdana" será la primera fuente de letra a utilizar.
- Font-stile: Estilo de la fuente de letra. Con el valor "normal" indicamos una fuente de letra normal.
- Font-size: Tamaño de la fuente de letra. En nuestro caso, el tamaño elegido es de 20 pixeles.
- Font-weight: Ancho de la fuente de letra. Se ha elegido el valor "Bold" (negrita) para resaltar el texto presentado.
- Color: Color de la fuente de letra. Se ha seleccionado el mismo color para este elemento HTML que el color de los escaques que representan las casillas negras.
- Line-height: Determina el espacio de interlineado. A mayor valor mayor espacio entre líneas. Se ha indicado el valor de 150% para ampliar el espacio entre el texto que determina el nombre de la posición y la línea de separación anterior al nombre.
- Text-align: Alineación del texto. Con el valor "left" mantenemos el nombre a la izquierda de la página HTML.
- Text-decoration: Decoración del texto. Con el valor "underline" el texto aparece subrayado.

El efecto conjunto sobre el elemento HTML es el siguiente:

Figura XXIII: Ejemplo de aplicación de estilos CSS.

NOTA: *En la sección de anexos de códigos fuente se incluye el contenido completo del fichero CSS.*

2.1.7 Análisis de programas de representación de partidas de ajedrez

La idea de crear una herramienta de tratamiento de ficheros PGN no es nueva. Quizás pueda restar puntos a la hora de evaluar la originalidad del PFC, pero debemos destacar que deja de manifiesto que se trata de una utilidad de gran interés por los seguidores de este "deporte-ciencia", por lo que sí conseguimos, al menos, un trabajo de gran utilidad. No obstante, sí podemos decir que existe originalidad en el enfoque realizado, dado que nuestro objetivo no se limita a la representación gráfica de una posición de una partida. Además, creamos una *base de conocimiento* que podría estar enlazada con la página HTML mediante elementos OWL, para que pueda ser utilizada por buscadores (semánticos) o mediante otras herramientas que estarán presentes en la inminente "Web 3.0". Este proyecto sólo se limita a la generación de contenido gráfico y semántico, pero podría ser la base para el desarrollo de un proyecto posterior donde se analicen las posibilidades de explotación de estos contenidos conjuntamente.

Como ya hemos comentado, existen muchas herramientas de tratamiento de ficheros PGN. Analicemos a continuación las más representativas dentro del mundo "open source", para tomar referencias a la hora de realizar el planteamiento de nuestro desarrollo.

Utilidad "Palmview"

Descripción: Aplicación de escritorio creada por Andrew Templeton, que genera contenido HTML con JavaScript embebido a partir de un fichero PGN. Tiene múltiples opciones de generación de dicho código, así como la generación de distintos tipos de página.

El aspecto del código HTML generado por esta aplicación es el siguiente:

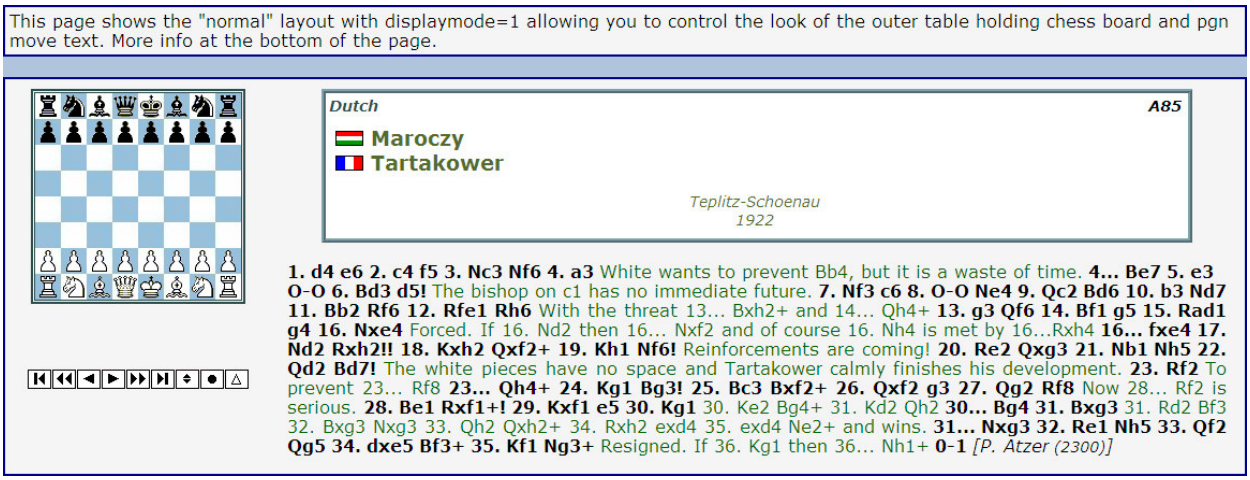
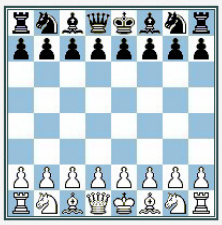


Figura XXIV: Aplicación Palmview.

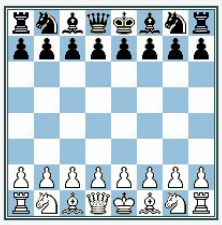


Gruenfeld **D94**

Rubinstein
Bogoljubow

Vienna
1922

1. d4 d5 2. c4 c6 3. e3 Nf6 4. Nc3 g6 5. Nf3 Bg7 6. Be2 O-O 7. O-O Nbd7? Now white gets control of the c-file. **8. cxd5! Nxd5 9. Nxd5 cxd5 10. Qb3 Nf6 11. Bd2 Ne4 12. Rfd1 Nxd2 13. Rxd2 Qd6 14. Rc1 b6 15. Rdc2 Bb7 16. Qa4** Prevents Rc8. **16... a6 17. Rc7 b5 18. Qa5 Rab8 19. R1c5 Rfd8 20. Ne5 Bf6** 20... Bxe5 21. dxe5 Qxe5 22. Rxb7± **21. Nc6 e6 22. g3** 22. Nxb8 Rxb8 followed by 23... Bd8 **22... Rdc8 23. Nxb8 Rxb8 24. Bxb5!** If now 24... axb5 then 25. Qa7 and wins. **24... Bd8 25. Be8 Qf8** Rubinstein would probably have enjoyed the endgame after 25... Bxc7 26. Qxc7 Qxc7 27. Rxc7 Rxe8 28. Rxb7 **26. Rxb7 Bxa5 27. Rxb8 Qd6 28. Rb7 Bb6 29. Rc6 Qb4 30. Bxf7+ 1-0**



Queen's Pawn **A47**

Carsten Hoi IM
Boris F Gulko GM

Olympiad
Salonika (Greece), 1988

1. d4 e6 2. Nf3 c5 3. e3 Nf6 4. Bd3 b6 5. O-O Bb7 6. Nbd2 cxd4 7. exd4 Be7 8. Re1 O-O 9. c3 d6 10. Qe2 Re8 11. Nf1 Nbd7 12. Ng3 Bf8 13. Bg5 h6?! 14. Bd2 Qc7 15. Bc2 Makes e6-e5 less dangerous and prepares the cannon 16. Qd3 etc. **15... Bd5 16. b3 Qb7** "Fasten seatbelts, and no smoking" as Hoi wrote in his comments. **17. Nh4!** Provokes a black advance on the king's side. **17... b5 18. Qd3** Now black must prevent 19. Nh5 followed by 20. Re3 and 21. Rg3 with a strong attack. **18... g5 19. Nf3 Bxf3?** Better 19... Bg7 **20. gxf3 Bg7 21. h4! gxh4 22. Ne4 Qc6 23. Kh1 Nh5** Threatens to win a piece with 24... f5 To continue the attack white must sacrifice. **24. Rg1! Kf8** Even if it looks very dangerous black should have played 24... f5 25. Bxh6 fxe4 26. Qe3! Re7! with a very messy position. After Kf8 it seems to be a forced win. **25. Rxg7! Kxg7** 25... Nxg7 26. Bxh6 **26. Bxh6+!** 26. Rg1+ Kf8 27. Bxh6+ Ke7 is unclear. **26... Kxh6 27. Rg1!** Blocks the escape route. **27... f5** 27... Nf4 28. Ng5!! f5 29. Qe3 e5 30. Nf7+ Kh7 31. Bxf5+ **28. Qe3+ f4** 28... Kh7 29. Qg5 **29. Nxd6!! Qxd6** 29... Ng3+ 30. Rxg3 fxe3 31. Nf7+ Kh5 32. Rg5# **30. Qd3 Nf8?** 30... Ng3+! 31. Rxg3 Nf8 32. Rg6+ Kh5 33. Rf6 Qe7 34. Rxf8 Qg7 35. Rxf4 Rf8 36. Rg4 Qf7 37. Kg2 and white wins due to the exposed black king. **31. Qh7+!!** Comments based on Carsten Hoi's in Skakbladet 1988. **1-0**

The chess piece set is based on the font Chess Merida by Armando Marroquin.

Notice that opening names and annotation symbols are tooltipped - when the mouse hovers a small text window appears with extended information. Playable null moves are used sometimes in the comments. For instance in the first game 13...Bxh2 and 14...Qh4 are replayed with no white moves in between.

This text (table) and the text at the page top was inserted by Palview from header and footer text files.

- Download CSS stylesheet
- Download ini file

Beware that URLs (links) in the files will need change if used for another site.

Como se puede ver, la misma página web muestra todas las partidas contenidas en el fichero PGN, de manera secuencial. En cada partida podemos ver todos los movimientos de la misma, y ofrece la posibilidad de seguimiento de movimientos mediante los botones de acciones: primer movimiento, movimiento anterior, etc.

Utilidad “pgn2html”

Descripción: Utilidad de escritorio creada por Werner Mueller. La principal funcionalidad de esta herramienta es la generación de código JavaScript válido para visionar una partida de ajedrez. La generación de código es bastante rápida y su interfaz es bastante interesante. Permite la generación de variantes de visualización y la inclusión de “flechas” de ayuda sobre el tablero, a modo de posibles planes de ejecución. El código resultante puede ser embebido en cualquier página web donde deseemos analizar dicha la partida.

Figura XXV: Aplicación pgn2html.

Utilidad "PGNtoJS"

Descripción: Utilidad de escritorio creada por Uwe Auerswald. Como la utilidad Palview, crea contenido HTML con JavaScript a partir de un fichero PGN.

Figura XXVI: Aplicación PGNtoJS.

Utilidad "LT-Pgn-Viewer"

Descripción: Utilidad web creada por Lutz Tautenhahn. Desde la página web que ofrece esta utilidad copiamos el contenido de un fichero PGN. Tras realizar esta acción, podemos analizar los movimientos de cualquier partida contenida en dicho fichero.

Figura XXVII: Aplicación LT-Pgn-Viewer.

Utilidad "PGN Reader Java Applet"

Descripción: Utilidad web creada por Kevin Coulombe. Un interesante Applet Java que nos permite ver partidas de ajedrez sobre una página web. La principal característica es que podemos pegar el contenido de un fichero PGN, en formato texto, y analizar a continuación cada una de las partidas contenidas.

Figura XXVIII: Aplicación PGN Reader Java Applet.

2.2 Implementación

En este segundo y último apartado de este capítulo se expondrá todo lo relacionado con la implementación realizada del generador **GPAHS**. Cada elemento de desarrollo será descrito en detalle en cada una de las secciones de este apartado. Los elementos que han intervenido en el programa desarrollado son los siguientes:

Contenido	Sección	Comentarios
Argumentos de invocación	2.2.1	En esta sección se describen los parámetros (argumentos) requeridos para realizar la invocación del proceso.
Analizador léxico	2.2.2	En esta sección se incluyen las decisiones de desarrollo relacionadas con el analizador léxico desarrollado con la herramienta LEX.
Analizador sintáctico	2.2.3	En esta sección se incluyen las decisiones de desarrollo relacionadas con el analizador sintáctico desarrollado con la herramienta CUP.
Analizador semántico	2.2.4	En esta sección se incluyen las principales reglas semánticas implementadas en el proceso.
Generación de código intermedio	2.2.5	En esta sección de presenta cómo se almacena en memoria la información requerida por las siguientes fases del proceso.
Optimización de código	2.2.6	En esta sección se presentan las principales acciones de mejora de código desarrolladas.
Generador de código	2.2.7	En esta sección se incluyen los comentarios relacionados con la parte del proceso responsable de la generación de páginas HTML.
Hoja de estilos	2.2.8	En esta sección se incluye la hoja de estilos que utilizarán las páginas HTML generadas.
Imágenes	2.2.9	En esta sección se presenta un inventario de todas las imágenes que han sido utilizadas en el desarrollo del generador.
Plantillas HTML y OWL	2.2.10	En esta sección se presenta la plantilla HTML de contenido estático utilizada como punto de partida para generar las páginas HTML.
Ficheros de procesamiento por lotes	2.2.11	En esta sección se presentan los ficheros de procesamiento por lotes utilizados para la compilación y pruebas de ejecución iniciales.
Manual de usuario	2.2.12	En esta sección se incluyen los comentarios que permiten ejecutar el programa generador GPAHS (guía de operación) y una presentación de la salida HTML generada (guía de usuario).

Tabla XLI: Secciones del apartado 2.2.

2.2.1 Argumentos de invocación

En esta sección presentamos los argumentos que deben ser utilizados a la hora de invocar al proceso **GPAHS**.

Argumento	Orden	Comentarios
Fichero PGN	1	El primer argumento incluye el nombre del fichero PGN a procesar. No se requiere un nuevo argumento del nombre HTML a generar dado que el nombre será el mismo pero con la extensión "html" en lugar de "pgn".
Plantilla HTML	2	El segundo argumento debe contener el nombre del fichero HTML que será utilizado como plantilla, con el contenido estático no dependiente del fichero PGN.
Plantilla OWL	3	El tercer argumento debe contener el nombre del fichero OWL (ontología) que será utilizada como plantilla inicial, con la definición de las clases, propiedades y facetas. Este contenido es estático y no depende del fichero PGN de entrada. Las instancias o individuos serán creados a partir de la información del fichero de entrada.
Táctica o estrategia	4	El tercer argumento debe contener el nombre de la táctica o estrategia relacionada con las posiciones del fichero PGN, a incluir en las instancias OWL contenidas en el fichero HTML. Se debe destacar que todas las posiciones contenidas en el fichero, por ello, deberán corresponder a una misma táctica o estrategia.
Traza	5	El cuarto y último argumento contendrá un string que determine si deseamos activar una traza del proceso. Con "true" se activará la traza y con "false" no se activará. Cualquier otro valor de argumento generará un error. En la traza presentamos el contenido del código intermedio generado, información útil para validar las fases léxica, sintáctica y semántica del proceso.

Tabla XLII: Argumentos de invocación.

La parte del programa java relacionado con la lectura y validación de estos argumentos se presenta a continuación.

El proceso debe validar, en primer lugar, que el número de argumentos incluidos en la invocación del programa es 5:

```
if (argv.length !=5) {
   System.out.println ("Error en la invocación del proceso. Uso:");
   System.out.println ("\tjava parser <argumento1> <argumento2> <argumento3> <argumento4>");
   System.out.println ("\tPrimer argumento ...: Fichero PGN a transformar, con extension \"pgn\".");
   System.out.println ("\tSegundo argumento ..: Plantilla HTML.");
   System.out.println ("\tTercer argumento ...: Plantilla OWL (ontologia).");
   System.out.println ("\tCuarto argumento ...: Tactica/Estrategia.");
   System.out.println ("\tQuinto argumento ...: Traza (true/false).");
   return;
}
```

El primer argumento, nombre del fichero PGN, debe contener la extensión "pgn" y debe ser el nombre de un fichero que exista en la carpeta desde donde estamos ejecutando el proceso **GPAHS**:

```
/* validación de formato y existencia de fichero */
   sFuente = argv[0];
   if (!sFuente.toUpperCase().endsWith(".PGN")) {
      System.out.println ("La extension .PGN es obligatoria en el fichero fuente");
      System.out.println ("Por favor, renombre el fichero " + sFuente + " y lance de nuevo el proceso");
   return;
   }
   fFuente = new File(sFuente);
   if (!fFuente.exists()) {
      System.out.println ("El fichero PGN no existe.");
      System.out.println ("Para poder ejecutar el proceso GPAHS se requiere un fichero PGN.");
      return;
   }
```

El segundo argumento, el nombre de la plantilla HTML, debe tener la extensión "html" y debe existir. En el caso de que sí exista preparamos los elementos java necesarios para realizar la lectura del fichero.

```
/* validación formato de fichero */
   sPlantillaHTML = argv[1];
   if (!sPlantillaHTML.toUpperCase().endsWith(".HTML")) {
      System.out.println ("La extension .HTML es obligatoria en el fichero plantilla");
      System.out.println ("Por favor, renombre el fichero " + sPlantillaHTML + " y lance de nuevo el proceso");
      return;
   }

/* Si el fichero plantilla no existe el proceso se cancela. */
      fPlantillaHTML = new File(sPlantillaHTML);
      if (!fPlantillaHTML.exists()) {
         System.out.println ("El fichero plantilla no existe.");
         System.out.println ("Para poder ejecutar el proceso GPAHS se requiere una plantilla HTML.");
         return;
      } else {
         frPlantillaHTML = new FileReader(sPlantillaHTML);
         brPlantillaHTML = new BufferedReader(frPlantillaHTML);
      }
```

El tercer argumento, el nombre de la plantilla OWL, debe tener la extensión "owl" y debe existir. En el caso de que sí exista preparamos los elementos java necesarios para realizar la lectura del fichero.

```
/* validación formato de fichero */
   sPlantillaOWL = argv[2];
   if (!sPlantillaOWL.toUpperCase().endsWith(".OWL")) {
      System.out.println ("La extension .OWL es obligatoria en el fichero plantilla");
      System.out.println ("Por favor, renombre el fichero " + sPlantillaOWL + " y lance de nuevo el proceso");
      return;
   }

/* Si el fichero plantilla no existe el proceso se cancela. */
      fPlantillaOWL = new File(sPlantillaOWL);
      if (!fPlantillaOWL.exists()) {
         System.out.println ("El fichero plantilla no existe.");
         System.out.println ("Para poder ejecutar el proceso GPAHS se requiere una plantilla OWL.");
         return;
      } else {
         frPlantillaOWL = new FileReader(sPlantillaOWL);
         brPlantillaOWL = new BufferedReader(frPlantillaOWL);
```

```
}
```

El cuarto argumento, el nombre de la táctica o estrategia asociada a las posiciones contenidas en el fichero PGN, debe ser un string válido, según la definición de la ontología realizada en el apartado anterior.

```
sPosicion = argv[3];
if (!sPosicion.equals("Position")        &&
   !sPosicion.equals("PositionBegin")    &&
   !sPosicion.equals("PositionMiddle")   &&
   !sPosicion.equals("PositionEnd")      &&
   !sPosicion.equals("OpeningError")     &&
   !sPosicion.equals("Trap")             &&
   !sPosicion.equals("Drowned")          &&
   !sPosicion.equals("ContinuousCheck")  &&
   !sPosicion.equals("Combination")      &&
   !sPosicion.equals("DoubleAttack")     &&
   !sPosicion.equals("DiscoveredAttack") &&
   !sPosicion.equals("DiscoveredCheck")  &&
   !sPosicion.equals("DoubleCheck")      &&
   !sPosicion.equals("Pinning")          &&
   !sPosicion.equals("Skewer")           &&
   !sPosicion.equals("Deflection")       &&
   !sPosicion.equals("Interception")     &&
   !sPosicion.equals("EliminateDefense") &&
   !sPosicion.equals("Freeing")          &&
   !sPosicion.equals("Lock")             &&
   !sPosicion.equals("Xray")             &&
   !sPosicion.equals("Overload")         &&
   !sPosicion.equals("WeakLine")         &&
   !sPosicion.equals("Zwischenzug")      &&
   !sPosicion.equals("Promotion")        &&
   !sPosicion.equals("AttackKingCenter") &&
   !sPosicion.equals("AttackCastledKing") &&
   !sPosicion.equals("PawnEndgame")      &&
   !sPosicion.equals("RookEndgame")      &&
   !sPosicion.equals("MinorPieceEndgame") &&
   !sPosicion.equals("QueenEndgame")     &&
   !sPosicion.equals("OtherEndgame")     &&
   !sPosicion.equals("MateInOne")        &&
   !sPosicion.equals("MateInTwo")        &&
   !sPosicion.equals("MateInThree")      &&
   !sPosicion.equals("OtherMate")) {
   System.out.println ("La posicion indicada en el argumento no es valida para la ontologia definida.");
   System.out.println ("Por favor, cambie " + sPosicion + " y lance de nuevo el proceso.");
   return;
}
```

En el quinto y último argumento indicamos el tipo de traza.

```
sTraza = argv[4];
if (!sTraza.equals("true") &&
   !sTraza.equals("false")) {
   System.out.println ("El valor de traza debe ser \"true\" o \"false\".");
   System.out.println ("Por favor, cambie " + sTraza + " y lance de nuevo el proceso.");
   return;
}
```

2.2.2 Analizador léxico

En esta sección incluimos las decisiones de desarrollo adoptadas con el componente relacionado con el analizador léxico que, como vimos, se realizará con la herramienta "Jlex".

Un fichero de especificación léxica Jlex se organiza en tres secciones, separadas por la directiva "%%" (tienen que ser los primeros caracteres de la línea. El contenido de cada sección es el siguiente:

- Sección de código de usuario. Es la primera sección del fichero de especificaciones léxicas, y se copia tal cual en el fichero java a generar con la herramienta Jlex. Esta sección se utiliza para incluir sentencias de importación, definición de clases y definición de tipos de datos que puedan ser requeridos por el proceso. Como veremos a continuación, esta sección ha sido utilizada en nuestro proceso.
- Sección de directivas: Es la siguiente sección del fichero, y se utiliza para particularizar algunas características del analizador léxico. También en esta sección se declaran las macros y estados que se usarán en la definición de reglas léxicas.
- Sección de reglas léxicas. En esta última sección se declaran las reglas léxicas que serán utilizadas por el analizador para determinar los token que deben ser devueltos al programa invocador del analizador léxico.

A continuación veremos cómo han sido planteadas y desarrolladas cada una de estas secciones.

Sección código de usuario

En nuestro caso, en esta sección se han incluido las sentencias de importación y las definiciones de clases requeridas por el proceso.

Las sentencias de importación incluidas han sido las siguientes:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java_cup.runtime.Symbol;
```

Igualmente, se ha definido la clase *Game* que permitirá contener toda la información relacionada con una posición determinada. Además, se ha definido la clase *TablaGames* para poder almacenar la información de todas las posiciones de un fichero PGN, dado que es una clase que extiende la clase estándar "LinkedHashMap". Esta clase permitirá almacenar toda la información del código intermedio del generador. Además, se ha definido el método "ShowContenido()", el cual permite obtener por consola el contenido de todas las posiciones almacenadas en la clase. Este método será invocado si el argumento de traza de invocación al proceso tiene el valor "true".

La definición de estas clases es la siguiente:

```
class Game {
   String event;
   String site;
…
   String move;
```

```
    public Game () {
       this.event = null;
       this.site = null;
…
       this.move = null;
    }
}

class TablaGames extends LinkedHashMap {

 public void ShowContenido() {
   Enumeration en;
   System.out.println("");
   Iterator it = this.keySet().iterator();
   while (it.hasNext()) {
     String key = (String) it.next();
     Game gm = (Game) this.get(key);
     System.out.println ("Event .....: " + gm.event);
     System.out.println ("Site .....: " + gm.site);
…
     System.out.println ("Move .....: " + gm.move);
     System.out.println ("");
   }
 }
}
```

Sección de directivas

En la sección de directivas se ha incluido la siguiente información:

- Declaración de macros, que serán utilizadas (sustituidas) en las definiciones de reglas léxicas que veremos a continuación. La declaración de macros permite mejorar la legibilidad de las reglas léxicas. Por cada regla léxica definida con una macro se realizará una sustitución de la expresión regular asociada a la macro. El límite de la definición de una expresión léxica se alcanza con el primer espacio en blanco que aparezca tras la definición.

Para mejorar la legibilidad de nuestro analizador nos apoyaremos en las siguientes definiciones de macros:

Macro	Expresión regular	Significado
cadenastring	"\""[^\n"\""]*"\""	Cadena "string", incluyendo los caracteres de doble comilla ("), inicial y final. Ejemplo: "Madrid"
string	[a-zA-Z0-9_\-\.\:\/\!\;\,\=\$\#\{\{\?\'\%]+	Cadena "string" sin dobles comillas.
move	[rnbqkpRNBQKPabcdefg]"x"?[abcdefgh][12345678]("+"|"#")?	Movimientos de la partida o solución de la posición propuesta.
Digitos	[0-9]+	Combinación de dígitos numéricos. Ejemplo: 435673
signo	"+"|"-"	El signo más (+) o el signo menos (-). Ejemplo: -
entero	{signo}?{digitos}	Número entero, compuesto de un signo y dígitos.

		Ejemplo: +34
nag	"$"{digitos}	Código NAG, compuesto del símbolo "dólar" ($) y dígitos. Ejemplo: $23

Tabla XLIII: Analizador léxico, macros.

El código resultante es el siguiente:

```
cadenastring = "\""[^\n"\""]*"\""
string = [a-zA-Z0-9_\-\.\:\/\!\;\,\=\$\#\{\{\?\'\%]+
move = [rnbqkpRNBQKPabcdefg]"x"?[abcdefgh][12345678]("+"|"#")?
digitos = [0-9]+
signo = "+"|"-"
entero = {signo}?{digitos}
nag = "$"{digitos}
```

- Código añadido a la clase. En la sección de directivas también se puede incluir código de usuario que, en este caso, se corresponderá con código interno que permitirá obtener el número de la línea actual de tratamiento, invocando al método "getLine()".
-

```
%{
        public int getLine(){
                return yyline;
        }
%}
```

- Directivas sobre reglas léxicas. Las directivas incluidas en esta sección han sido las siguientes:
    - Declaración de estados. Hay caracteres, como el "corchete" ([) que puede aparecer tanto en la primera sección de una partida de un fichero PGN, como en la información que determina los movimientos realizados en la partida. Para evitar la ambigüedad se ha definido el estado "MOVE", que estará activo cuando el analizador léxico esté tratando la información relacionada con los movimientos realizados. En la sección de pares de etiquetas, el estado activo será el inicial: "YYINITIAL".
    - Activación de valores léxicos. Para que los valores léxicos "yychar" y "yyline" estén disponibles en las acciones léxicas hace falta activarlos con las directivas siguientes: "%char" y "%line" respectivamente.
    - Formatos de fichero y texto. Para ampliar el alfabeto ASCII e incluir todos los caracteres de 8 bits (Extended ASCII), comprendidos entre el 0 (null) y el 255, se debe utilizar la directiva "%full". Para asegurar la independencia del programa generado respecto al tratamiento de los caracteres de salto de línea entre los sistemas UNIX y los basados en DOS, se incluirá también la directiva "%notunix".
    - Compatibilidad CUP. Para que Jlex y CUP sean compatibles y desde este último pueda tratar correctamente los tokens proporcionados por el primero, se requiere incluir la directiva "%cup".
    - Directivas modificadoras de la clase. Por último, para modificar el nombre de la clase generada por Jlex, incluiremos la directiva "%class" para renombrar la clase como "lexGPAHS".

○

```
%state MOVE
%char
%line
%full
%notunix
%cup
%class lexGPAHS
```

Sección de reglas léxicas

La tercera y última sección del fichero de especificación contiene las reglas léxicas que nos permitirán convertir el fichero de entrada en una secuencia de testigos o tokens. Cada una de éstas se estructura en tres partes:

- Lista opcional de estados.
- Expresiones regulares.
- Acciones léxicas.

Lista opcional de estados

En la especificación léxica de un lenguaje nos podemos encontrar con que varios de sus componentes tienen diferentes significados o interpretaciones en función del lugar donde aparezcan dentro del fichero de entrada. Esta circunstancia ocurre, por ejemplo, con el token “abrir corchete” ([), dado que aparece en la primera sección de pares “etiqueta-valor” y es un símbolo válido en la especificación de movimientos.

Los estados que serán utilizados, como ya se ha indicado, son los siguientes:

- YYINITIAL: Estado inicial, activo en el tratamiento de los pares “etiqueta-valor”.
- MOVE: En el tratamiento de los movimientos en notación algebraica.

Expresiones regulares y Acciones léxicas

Las expresiones regulares serán definidas bien directamente o a través de una macro. Las acciones léxicas son el código Java que introducimos para que se ejecute cuando una entrada del fichero a tratar se corresponde con la regla léxica (expresión regular) que precede a esta acción, expresión definida directamente o mediante macros. En general, nuestro analizador léxico devolverá tanto el símbolo que deseamos asociar a cada expresión regular como el valor encontrado.

Algunas de las expresiones regulares y acciones léxicas incluidas en nuestro generador son las siguientes:

```
<YYINITIAL>[\t \r\n]*              { /* ignorar blancos */
                }
…
<YYINITIAL>{cadenastring}  { /* System.out.println("OK cadena..: " + yytext()); */
                return new Symbol(sym.TK_CADENASTRING, new String(yytext().replaceAll("\"", "")));
                }
…
<MOVE>"1-0"       {
            yybegin (YYINITIAL);
```

```
            return new Symbol(sym.TK_WIN);
            }
…
<MOVE>.         {
            int line = yyline+1;
            System.out.println("Error lexico en linea " + line + " - " + "Simbolo desconocido en MOVE: " + yytext());
            }
```

NOTA: *En la sección de anexos se incluye el código fuente del fichero completo Jlex.*

2.2.3 Analizador sintáctico

En esta sección analizaremos los componentes relacionados con el análisis sintáctico a realizar del fichero PGN de entrada. El proceso de análisis sintáctico será responsable de obtener toda la información asociada a cada etiqueta de la sección inicial de cada partida, así como el conjunto de movimientos y comentarios incluidos en la segunda sección. El procedimiento será el siguiente:

Mientras no se alcance el final del fichero:

Mientras no se alcance el final de una partida (posición), hacer:

Si es un valor asociado a una etiqueta:

Almacenar valor a un atributo temporal.

Si forma parte del conjunto de movimientos de una partida:

Concatenar la información obtenida con la obtenida en tokens anteriores.

Una vez obtenida toda la información de una partida:

Aplicar reglas semánticas

Almacenar información en la tabla de código intermedio

Limpiar contenido de atributos intermedios.

NOTA: *La funcionalidad asociada a la aplicación de reglas semánticas será descrita en la siguiente sección del documento*.

Comencemos presentando en primer lugar la definición final de la gramática PGN, para describir a continuación cada uno de los métodos utilizados en dicha gramática. En los ficheros CUP, la gramática se define mediante elementos terminales, elementos no terminales y la gramática propiamente dicha.

Ejemplo de elementos terminales:

```
terminal TK_CORCHETE_ABRIR, TK_CORCHETE_CERRAR;
…
terminal TK_TIMECONTROL, TK_SETUP, TK_FEN, TK_TERMINATION, TK_ANNOTATOR, TK_MODE, TK_PLYCOUNT;
```

Ejemplo de elementos no terminales:

```
non terminal PGN_database, PGN_game, tag_section, movetext_section, tag_pair;
…
non terminal element, element_sequence, game_termination;
```

A continuación incluimos algunas secciones de la gramática definida:

```
PGN_database::=
        | PGN_game PGN_database
        | error  {: parser.errors.add("Error en linea " + parser.scanner.getLine() + " - " + "Error de formato PGN"); :}
```

```
                        ;
…
tag_pair::=     pair_event
          | pair_site
  …
          | error  {: parser.errors.add("Error en linea " + parser.scanner.getLine() + " - " + "Identificador de etiqueta no definido"); :}
          ;

pair_event::=     TK_CORCHETE_ABRIR TK_EVENT TK_CADENASTRING:s TK_CORCHETE_CERRAR
              {:
              parser.event = s.toString();
              :};
…
game_termination ::=     TK_WIN
                {:
                parser.id += 1;
                parser.CrearGame(parser.id);
                parser.LimpiarGame();
                :}
…
```

Para mejorar la legibilidad de la gramática, se ha recurrido a la utilización de métodos auxiliares que permiten realizar las acciones correspondientes al elemento sintáctico detectado, siempre que dicha acción haya requerido algo más complejo que el simple movimiento del *string* obtenido a un atributo temporal.

El método "IncluirNAG()" permite convertir el código NAG contenido en el fichero PGN (139 posibles códigos) en el texto o símbolo asociado a dicho código. Por ejemplo, el código NAG "$1" se corresponde con un buen movimiento, que en la mayoría de los programas visores de ficheros PGN se representa por el símbolo de "exclamación" (!). Esta información se concatena con la información ya almacenada temporalmente de movimientos de una partida. A continuación mostramos un fragmento del contenido del método:

```
  public static void IncluirNAG(String nag) {

    String temp = null;
    if     (nag.equals("$1"))   temp = "!";
    else if (nag.equals("$2"))   temp = "?";
…
    else if (nag.equals("$138")) temp = "White has severe time control pressure";
    else if (nag.equals("$139")) temp = "Black has severe time control pressure";
    else temp = nag;
    if (parser.move == null) {
      parser.move = temp;
    } else {
      parser.move = parser.move.concat(temp);
    }
  }
```

Cuando el analizador detecta el final de movimientos de una partida (ganada, perdida, tablas o resultado desconocido), se invocan dos métodos. El primero, "CrearGame()", permite almacenar en una tabla intermedia toda la información relacionada con una partida (posición) determinada.

```
  public static void CrearGame(int id) {
    String numCadena = String.valueOf(id);
    Game gm = new Game();
    gm.event = event;
```

```
      gm.site = site;
…
      gm.move = move;
      parser.games.put(numCadena, gm);
    }
```

El segundo método, “LimpiarGame()”, inicializa a nulos todos los atributos de almacenamiento temporal.

```
    public static void LimpiarGame() {
      event = null;
      site = null;
…
      move = null;
    }
```

NOTA: *En la sección de anexos se incluye el código fuente completo del fichero CUP.*

2.2.4 Analizador semántico

En esta sección incluimos la descripción de los elementos de desarrollo relacionados con el tratamiento semántico del proceso. En particular, solo se ha requerido desarrollar un método que será invocado antes de guardar la información de una partida en la tabla de código intermedio para realizar el análisis semántico de la información. Esta fase del generador será la responsable de eliminar toda aquella información que no sea relevante en la salida HTML a generar.

Por ejemplo, y como veíamos en la fase de diseño, los campos obligatorios de etiquetas, cuando no se conoce el valor asociado a dicha etiqueta, se añade el carácter especial "?". Este dato no es relevante y, por ello, no será incluido en la salida de la página HTML o contenido de la ontología. En general, esta fase del proceso será la responsable de eliminar todo aquel dato que no aporte información significativa.

A continuación mostramos un fragmento del método Semantica(), responsable de esta depuración semántica:

```
   public static void Semantica()   {

/* event */
      if (event.equals("?")) event = null;
/* site */
      if (site.equals("?")) site = null;
…
/* move */
      if (move != null) {
         if (move.equals("?")) move = null;
      }
    }
```

2.2.5 Generación de código intermedio

El código intermedio se corresponde con la información almacenada temporalmente en memoria, elaborada durante la fase de análisis y utilizada en la fase de síntesis de nuestro generador. Todo el desarrollo relacionado con esta funcionalidad se muestra en esta sección.

La información temporal, posiciones de un fichero PGN, será almacenada en un objeto de la clase "*LinkedHashMap()*", subclase de la clase "*HashMap()*". En la sección de optimización de código indicamos el motivo de uso de esta clase.

```
class TablaGames extends LinkedHashMap { … }…
```

El objeto *games,* de la clase anterior, permitirá almacenar toda la información temporal.

```
static TablaGames games = new TablaGames();
```

En el método "*crearGame()*" almacenamos la información de una posición sobre la tabla de todas las posiciones:

```
parser.games.put(numCadena, gm);
```

En la fase de síntesis y a través de un Iterator, recorreremos el contenido de la tabla anterior para generar el código HTML dinámico y las instancias OWL:

```
        Iterator it = games.keySet().iterator();
        while (it.hasNext()) {
          indice += 1;
          String key = (String) it.next();
          Game gm = (Game) games.get(key);
…
```

En el caso de que la traza esté activa, se invocará al método "*ShowContenido()*" para mostrar todo el contenido de la memoria intermedia:

```
if (sTraza.equals("true")) {
   parser.games.ShowContenido();
}
```

2.2.6 Optimizador de código

Tras la primera versión del generador, sobre el código se han realizado varias depuraciones para mejorar su legibilidad y para mejorar también el rendimiento y características del código generado. En esta sección sólo incluiremos las mejoras estrictamente relacionadas con la optimización de código, por lo que no se incluyen las mejoras de legibilidad y estructuración.

Ordenación de posiciones

En la primera versión del programa se utilizó una clase "*HashTable()*" para almacenar la información del código intermedio.

```
class TablaGames extends Hashtable { … }
```

Las acciones que se realizaban con este tipo de tabla son similares a las comentadas en la sección anterior. El problema que nos encontramos al utilizar esta clase es que la recuperación de la información se correspondía con la forma de actuar de las colas LIFO (Last In First Out). Es decir, la última posición recuperada del fichero PGN era la primera en ser tratada en la fase de síntesis y, por ello, la primera en aparecer en las páginas HTML y base de conocimiento. Para evitar esta salida inversa, se modificó la clase "*Hashtable()*" por una subclase de "*HashMap()*", la cual permite poder obtener la información de forma ordenada, siguiendo el comportamiento de las colas FIFO (Fisrt In First Out).

Contenido estático y dinámico

En la primera versión del programa (no completo) se generaba todo el código HTML de forma automática; sin utilizar, por lo tanto, ninguna plantilla HTML. Realizar el proceso de síntesis de esta forma tiene dos claros inconvenientes:

- El mantenimiento del código Java es más engorroso, dado que se requieren más sentencias de generación de código.
- El rendimiento del proceso es menor.

Por ello, se realizó una adaptación del proceso que consistía en apoyarse en una plantilla de código HTML donde se incluye todo el código estático de la página HTML, dejando exclusivamente a la fase de síntesis la generación dinámica de información relacionada con los datos obtenidos del fichero fuente PGN.

Utilizando esta misma lógica, se parte de un fichero OWL con la definición de clases y propiedades, y se incluye dinámicamente sobre el fichero OWL generado las instancias, contenido dinámico obtenido a partir de los datos del fichero fuente PGN, como en el caso anterior.

NOTA: *En la siguiente sección se detalla este proceso de generación de código.*

Codificación UTF-8

En la primera versión del programa se utilizaba la clase "PrintWriter" para generar las entradas de los ficheros de salida, tanto del contenido HTML como del contenido OWL. Esta clase no permite especificar el tipo de codificación (encoding) de los caracteres generados que, por defecto, se corresponden con una codificación no-UTF-8. Protégé, en la versión utilizada en este PFC, solo funciona con la codificación UTF-8, por lo que algunos ficheros OWL generados, si el fichero fuente contenía caracteres especiales (ê, ü, etc.) no se podían abrir correctamente, con generación de excepciones Java no controladas por el proceso.

Para solucionar el problema se sustituyó esta clase por la clase "BufferedWriter", con apoyo en la clase "OutputStreamWriter".

```
String sDestinoOWL = null;
File fDestinoOWL = null;
FileOutputStream fosDestinoOWL = null;
OutputStreamWriter oswDestinoOWL = null;
BufferedWriter bwDestinoOWL = null;
```

La ventaja de utilizar estas nuevas clases es que se permite especificar el tipo de codificación a utilizar.

```
fosDestinoOWL = new FileOutputStream(sDestinoOWL);
oswDestinoOWL = new OutputStreamWriter(fosDestinoOWL, "UTF-8");
bwDestinoOWL = new BufferedWriter(oswDestinoOWL);
```

Tras realizar este cambio, todos los ficheros OWL generados se pueden abrir con la herramienta Protégé.

Generación de código HTML y OWL conjunta Vs independiente

En la primera versión del programa generador se realizó un único tratamiento de los datos en memoria, generando en paralelo tanto la información HTML como OWL. Esta versión del programa es válida, incluso más eficiente desde el punto de vista de rendimiento que la presentada en esta memoria, pero el código fuente era poco legible, por lo que el mantenimiento hubiese sido muy complicado.

En su lugar, se han desarrollado dos procesos completamente independientes, por ello, se requieren realizar dos bucles de tratamiento de la información almacenada en el código intermedio.

Bucle de tratamiento de código intermedio:

Generación de código HTML.

Bucle de tratamiento de código intermedio:

Generación de código OWL.

NOTA: *También se podría haber creado un único fichero de salida HTML con el contenido de las instancias OWL correspondientes. Esta versión del proceso era la que se consideró al iniciar el enfoque del PFC, pero la versión final presentada es mucho más versátil, dado que independiza el contenido HTML puro, de la base de conocimiento.*

Instancias OWL con el carácter "&" en contenido

Algunas instancias OWL contenían el carácter "&" como parte del texto de algunos valores de propiedades. En estos casos, los ficheros OWL no se podían abrir, generándose una excepción Java tras el intento de apertura. Para solución este problema, se ha creado un método en el generador, dentro de la fase de depuración semántica, que sustituye el carácter "&" por el string "and".

Nuevos tags PGN

Durante la fase de análisis y diseño se incluyeron en la definición de la ontología todos los tags de etiquetas PGN presentes en el documento oficial que normaliza su contenido. No obstante, durante la fase de pruebas se han encontrado algunas etiquetas adicionales no normalizadas. Con estas nuevas etiquetas se han seguido los siguientes criterios en función de los siguientes condicionantes:

- Si la etiqueta se ha encontrado en más de un fichero PGN fuente, con un contenido que podría ser incluido en las próximas normalizaciones PGN:
    - Las nuevas etiquetas se han incluido en los analizadores léxicos / sintácticos, así como en la definición de la ontología. Etiquetas incluidas:
        - *eventType*: Tipo de evento.
        - *eventCountry*: Ciudad donde se desarrolla el evento.
        - *eventRounds*: Número de rondas en la que se disputa el evento.
- Las nuevas etiquetas sólo se han encontrado en un fichero PGN fuente, y no parecen ser etiquetas que vayan a ser incluidas en las próximas normalizaciones PGN:
    - Las nuevas etiquetas se han incluido en los analizadores léxicos / sintácticos, se presentan en la página HTML como información complementaria pero no se incluyen en la definición de la ontología. Etiquetas no incluidas en la ontología:
        - *emaWhite*: Significado desconocido.
        - *emaBlack*: Significado desconocido.
- Las nuevas etiquetas sólo se han encontrado en un fichero PGN y en muy pocas posiciones:
    - Las nuevas etiquetas se han eliminado del fichero PGN fuente.

NOTA: *Las únicas etiquetas encontradas en este último caso parecen estar muy relacionadas con características de los productos del fabricante de software ChessBase*.

2.2.7 Generador de código

En esta sección expondremos los mecanismos desarrollados para generar el código HTML y OWL, objetivo básico de nuestro PFC.

Generación de código HTML

Como ya se ha indicado en la sección anterior, el código HTML final se genera a partir de la información contenida en una plantilla HTML (datos estáticos) y de la información obtenida del fichero fuente PGN (datos dinámicos). La plantilla HTML contiene 2 strings a modo de comentario que son utilizados por el generador para determinar los puntos donde se debe realizar la inclusión de código dinámico. En el primer punto se incluye la declaración de las variables JavaScript que hacen referencia a cada uno de los elementos *canvas* definidos en la página. Elemento de control:

```
//<GPAHS_INSERT_1>
```

Ejemplo de inserción de código:

```
// EJEMPLO DE ENTRADAS A GENERAR POR GPAHS:
// posicion 1:
// var ctx1 = document.getElementById("canvas1").getContext('2d');
// posicion(ctx1, "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq e7 0 1");
// posicion 2:
// var ctx2 = document.getElementById("canvas2").getContext('2d');
```

En el segundo punto de control debemos incluir el código que inserta los elementos HTML que permiten definir el elemento "*canvas*", y la inclusión de información complementaria de la posición. Elemento de control:

```
<!--<GPAHS_INSERT_2>-->
```

Ejemplo de inserción de código:

```
<!-- EJEMPLO DE ENTRADAS A GENERAR POR GPAHS:                    -->
<!--   <div class="posicion">                                    -->
<!--     <p class="posicion_nombre">Posición 1</p>               -->
<!--     <table border=".5">                                     -->
<!--     <tr><td>                                                -->
<!--     <p class="posicion_gráfico">                            -->
<!--       <canvas id="canvas1" width="390" height="300">        -->
<!--       Su browser no acepta elementos canvas del HTML5.      -->
<!--       </canvas>                                             -->
<!--     </p></td>                                               -->
<!--     <td valign="top" align="right">Texto de la derecha, en el top</td>  -->
<!--     </table>                                                -->
<!--     <p class="posicion_solucion">solución ...</p>           -->
<!--     <ul class="actions">                                    -->
<!--       <li class="solucion"><a href="" title="Solución">Solución</a></li>  -->
```

```
<!--     </ul>                                    -->
<!--   </div> -->
<!--   <hr /> -->
```

El procedimiento seguido en esta fase del generador es el siguiente:

Mientras no se alcance el final del fichero plantilla HTML:

Si hemos alcanzado el bloque 1 de control:

Incluimos todas las sentencias JavaScript correspondientes a este bloque

(bucle de lectura del código intermedio)

Si hemos alcanzado el bloque 2 de control:

Incluimos todas las sentencias HTML correspondientes a este bloque

(nuevo bucle de lectura del código intermedio)

Si no hemos alcanzado ni el bloque 1 ni el bloque 2:

Pasamos al fichero de salida la línea obtenida del fichero de entrada.

Control de errores y cierre de los ficheros de entrada y salida.

Generación de código OWL

En este segundo proceso debemos generar la base de conocimiento a partir de la plantilla OWL estática, que contiene la definición de clases y propiedades, y la información dinámica obtenida del código intermedio.

En este caso, el procedimiento de creación del fichero final OWL es mucho más simple:

Mientras no se alcance el final del fichero plantilla OWL:

Pasamos al fichero de salida la línea obtenida del fichero de entrada (clases y propiedades) .

Bucle de lectura del código intermedio:

Pasamos al fichero de salida los datos obtenidos del código intermedio (instancias).

Control de errores y cierre de los ficheros de entrada y salida.

NOTA: *Todo el código relacionado con el proceso de generación se encuentra contenido en el fichero "cupGPAHS.cup". En el apartado de anexos de código fuente se encuentra integro este fichero.*

2.2.8 Hoja de estilos

La hoja de estilos que ha sido implementada para dar formato a la página HTML incluye los siguientes elementos:

Elemento "body" (cuerpo HTML): estilo genérico para determinar el ancho de la página en píxeles y el margen izquierdo.

```
body{
  width: 960px;
  margin: 15px auto;
}
```

Elemento "p" (párrafo): estilo utilizado para mostrar la solución de la posición didáctica propuesta.

```
p{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:18px;
  font-weight:normal;
  color:rgb(0,0,0);
  line-height: 100%;
  text-align:left;
}
```

Elemento "li" (lista de elementos desordenados): Se ha utilizado en la página HTML para incluir el link de solución.

```
li{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:18px;
  font-weight:normal;
  color:rgb(204,204,153);
  line-height: 100%;
  text-align:left;
}
```

Elemento "hr" (línea horizontal): estilo utilizado para indicar el color de las líneas de separación de posiciones.

```
hr {
  color:rgb(204,204,153);
}
```

Elemento clase "posición_nombre" de párrafo: Para dar un estilo particular al párrafo que contiene el nombre de la posición (…).

```
p.posicion_nombre{
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:20px;
 font-weight:bold;
 color:rgb(204,204,153);
 line-height: 150%;
 text-align:left;
 text-decoration:underline;
}
```

Elemento id "data1": estilo que determina el formato del texto de la tabla de información complementaria de una posición.

```
#data1 {
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:12px;
 font-weight:bold;
 color:rgb(0,0,0);
 text-align:left;
 vertical-align:top;
 }
```

Elemento id "data2": En la tabla de información complementaria de una posición se incluye este nuevo estilo para que los títulos de cada campo presenten un estilo diferenciado.

```
#data2 {
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:12px;
 font-weight:bold;
 color:rgb(255,255,255);
 background-color:rgb(204,204,153);
 text-align:left;
 vertical-align:top;
 }
```

Elemento "h2": Estilo asociado a los títulos de tamaño "h2", que en nuestro caso se corresponde con el acrónimo de nuestro aplicativo: "GPAHS".

```
h2{
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:25px;
 font-weight:bold;
 color:rgb(204,204,153);
 line-height: 20px;
 text-align:center;
 }
```

Elemento "h3": Estilo asociado a los títulos de tamaño "h3", que en nuestro caso se corresponde con el nombre que hemos incluido en la plantilla HTML: "Chess Exercises".

```
h3{
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:25px;
 font-weight:bold;
 color:rgb(204,204,153);
 line-height: 20px;
 text-align:center;
 }
```

Obviamente, cualquier otra opción de diseño hubiese sido válida. Por ello, queda a las preferencias del lector la adaptación de esta hoja de estilos.

NOTA: *En los anexos de código fuente se incluye el código fuente completo de la hoja de estilos presentada.*

2.2.9 Imágenes

En esta sección se incluyen todas las imágenes que han sido utilizadas en el desarrollo del generador. Por un lado, está la imagen que aparece en la cabecera de las páginas HTML generadas (que coincide con el también utilizado en la primera página de este documento), y las figuras utilizadas para dibujar las posiciones de ajedrez. Aunque sólo se han utilizado las figuras de resolución "30x30" píxeles, dado que cada escaque del tablero ha sido definido con este tamaño, también se entrega junto a la memoria de este PFC otras piezas de otro tamaño, para poder ser utilizadas en nuevas versiones del programa.

Figura XXIX: Logo GPAHS.

El logo del aplicativo **GPAHS** es el que aparece en el párrafo anterior, y las posibles piezas a utilizar son las siguientes:

Tamaño	Bando	Rey	Dama	Torre	Alfil	Caballo	Peón
30*30	Blancas						
	Negras						
29*29	Blancas						
	Negras						
28*28	Blancas						
	Negras						
27*27	Blancas						
	Negras						
26*26	Blancas						
	Negras						
25*25	Blancas						
	Negras						
24*24	Blancas						
	Negras						
23*23	Blancas						
	Negras						
22*22	Blancas						
	Negras						
21*21	Blancas						
	Negras						
20*20	Blancas						
	Negras						

Tabla XLIV: Imágenes de piezas.

2.2.10 Plantillas HTML y OWL

Como ya se ha indicado en las secciones anteriores, el código estático de las páginas HTML a generar, y la definición de clases y propiedades de nuestra ontología serán obtenidos a partir dos ficheros que serán tratados por el proceso como plantillas.

El fichero "*GPAHSPlantilla.html*" se corresponde con la plantilla HTML. Veamos a continuación un resumen de los elementos principales contenidos en dicha plantilla:

```
…

<!DOCTYPE html>
<html lang="es">
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>GPAHS</title>
   <link rel="stylesheet" href="css/style.css" type="text/css">

<!-- JavaScript de JQuery: -->
   <script type="text/javascript"
         charset="utf-8"
         src="js/jquery-1.5.2.js">
   </script>

<!-- JavaScript que genera el dinamismo de la pagina, utilizando Jquery: -->
   <script type="text/javascript"
         charset="utf-8"
         src="js/GPAHS.js">
   </script>

<!-- JavaScript que genera el tablero: -->
   <script>

var fila = new Number();
var columna = new Number();
…
var bk = new Image(); // rey negro
..
var wp = new Image(); // peon blanco
…
bk.src = "img/30/bk.png";
…
wp.src = "img/30/wp.png";

function init() {
…
//<GPAHS_INSERT_1>
}

function posicion(ctx, fen) {
…
// dibujar el tablero:
…
// dibujar texto de filas y columnas:
…
//tratamiento del campo FEN:
…
// Campo 1.- Posición de las piezas en el tablero:
…
// Campo 2.- Turno de juego:
…
 Campo 3.- Posibilidad de enroque:
…
// Campo 4.- Captura al paso:
```

```
…
// Campo 5.- Movimientos sin captura:
…
// Campo 6.- Movimientos de partida:
…

function pixel(f, c) {
…
}

function rads(x) { return Math.PI*x/180 }
…
</script>

 </head>
 <body onLoad="init();">
 <header id="page_header">
   <table>
     <tr><td rowspan="2"><img src="img/chess.gif" alt="Logo GPAHS" /></td>
       <td ><h2>GPAHS</h2></td>
     </tr>
     <tr>
       <td><h3>Chess Exercises</h3></td>
     </tr>
   </table>
 </header>

 <div id="container">
<!--<GPAHS_INSERT_2>-->
 </div>

 <footer id="page_footer">
   <hr />
   <p>GPAHS - Generador de Posiciones de Ajedrez en HTML5 Semántico</p>
   <p>&copy; 2011 UOC</p>
 </footer>
 </body>

</html>
```

Contenido en este fichero se encuentra el código JavaScript que permite representar gráficamente una posición de ajedrez, así como el resto de información relacionada con dicha posición (movimientos sin captura, enroques posibles, etc.).

El fichero "*ChessPosition.owl*" se corresponde con la plantilla OWL, y es el fichero directamente elaborado con la herramienta Protégé. Tómese como referencia de contenido lo mostrado en la sección 2.5.1, "Diseño con Protégé".

2.2.11 Ficheros de procesamiento por lotes

Para poder mejorar el rendimiento de trabajo durante el desarrollo del PFC, se han creado cuatro ficheros de procesamiento por lotes con distinta finalidad:

Fichero	Finalidad	Comentarios
PFC_compilar.bat	Compilación	Este fichero permite compilar todos los objetos requeridos para poder ejecutar el generador GPAHS. Los pasos realizados son los siguientes: • Verificación de la versión de la máquina virtual java. • Borrado de los ficheros temporales "java", creados como ficheros intermedios durante el proceso de compilación. • Generación del fichero java con la herramienta Jlex. • Renombrado del fichero creado en la fase anterior. • Generación del fichero java con la herramienta CUP. • Compilación de todos los ficheros java (parser, sym y lexGPAHS). NOTA: *En la versión final del fichero se mantiene la compilación con el parámetro "-Xlint:unchecked", lo cual permite presentar los mensajes relacionados con la verificación de conversión de tipos en invocaciones a métodos dentro del programa. Un mensaje de advertencia aparece en el proceso de compilación, pero no se ha realizado ninguna acción correctiva por estar controlados los efectos en la invocación al método "put" para guardar en memoria intermedia toda la información relacionada con una posición. No existe posibilidad de errores en tiempo de ejecución.*
PFC_ejecutar_unitaria.bat	Verificación inicial	Este fichero permite ejecutar el proceso GPAHS con un fichero de entrada con una única posición pero con contenido en todas las etiquetas válidas del formato PGN. La página HTML generada contiene todos los posibles datos asociados a la posición, y el individuo OWL contiene igualmente toda la información asociada.
PFC_ejecutar_verificacion.bat	Verificación errores	Este fichero permite validar distintas circunstancias de error que el proceso debe detectar y actuar en consecuencia.
PFC_ejecutar_integrada.bat	Pruebas integradas	Por último, este fichero contiene una batería de pruebas que permite asegurar el correcto funcionamiento del generador.

Tabla XLV: Ficheros de procesamiento por lotes.

NOTA: *El fuente de estos ficheros aparece en el apartado de anexos de código fuente*.

2.2.12 Manual de usuario

En esta sección se incluye una breve descripción del proceso de ejecución del generador, así como la descripción de los elementos que aparecen en la página HTML.

Proceso de ejecución del generador

El generador GPAHS presentado conjuntamente con esta memoria es una aplicación Java de consola, que se ejecuta con el siguiente comando:

```
java -cp CUP_10k.jar;. parser Prueba1.pgn GPAHSPlantilla.html ChessPosition.owl Position false
```

Como se puede ver, la ejecución del programa "parser" se realiza con el apoyo de la librería CUP (CUP_10k.jar) y se requiere incluir en la invocación 5 parámetros:

- Nombre del fichero PGN fuente. El fichero debe estar ubicado en la carpeta "pgn" y no se debe incluir en el nombre del fichero el path de acceso.
- Nombre de la plantilla HTML. El fichero plantilla HTML (datos estáticos) debe estar ubicado en la misma carpeta raíz donde se encuentre el programa *parser*.
- Nombre de la plantilla OWL. El fichero plantilla OWL (clases y propiedades) debe estar ubicado en la misma carpeta raíz donde se encuentre el programa *parser*.
- Clase: Nombre de la clase OWL que determina la situación táctica/estratégica presente en el fichero fuente.
- Traza: Tipo de traza. "true" si deseamos mostrar por consola el contenido de la memoria intermedia. "false" en caso contrario.

El proceso verifica, por razones de seguridad, que los ficheros destino HTML y OWL no existen en sus subcarpetas correspondientes ("html" y "owl" respectivamente). Por ello, antes de cada ejecución de un mismo fichero se deben borrar los ficheros creados en la ejecución anterior. Por ejemplo, para el comando presentado anteriormente, se deben ejecutar los comandos:

```
del .\html\Prueba1.html
del .\owl\Prueba1.owl
```

NOTA: *Para facilitar la ejecución de este proceso se presentan varios ficheros de procesamiento por lotes con los comandos correctos de ejecución.*

Página HTML generada

La página HTML generada presenta las siguientes secciones:

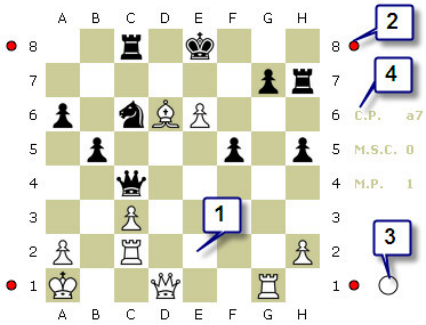
Sección	Comentarios
 GPAHS Chess Exercises	Cabecera de la página. Contenido estático.
Position 1	Por cada solución propuesta aparece un separador y un título ("Position 1", "Position 2", etc.).
	Posición didáctica, con las siguientes secciones: 1.- Tablero de ajedrez con las piezas ubicadas correctamente sobre el tablero. Para identificar mejor cada escaque se incluye el número de fila a la izquierda y derecha del tablero; y la letra de cada columna por encima y por debajo del tablero. 2.- Posibilidad de enroque. Un círculo en color rojo determina que no es posible el enroque y un círculo en color verde determina que sí es posible el enroque. En el ejemplo adjunto no es posible ningún enroque. 3.- Turno de juego. Un círculo blanco en la parte inferior derecha representa que deben mover las blancas, y un círculo negro en la parte superior derecha representa que deben mover las negras. En el ejemplo adjunto deben mover las blancas.
4.- Los valores que aparecen en esta sección del tablero son los siguientes: - "C.P." determina el peón que puede ser toma al paso. Si este indicador no aparece significa que no existe la posibilidad de tomar al paso. - "M.S.C" son los movimientos sin captura, que permiten determinar la proximidad de tablas por esta circunstancia. Cuando este valor no es significativo aparece el valor "0". Este elemento siempre aparece. - "M.P." determina los movimientos de partida. Cuando este valor no es significativo aparecerá el valor "1". Este elemento siempre aparece.	
Event ..: Becs Date ..: 1898 White ..: Tschigorin, Mikhail Black ..: Caro, H.. Result ..: 1-0 FEN ..: 2r1k3/6pr/p1nBP3/1p3p1p/2q5/2P5/P1R4P/K2Q2R1 w - a7 0 1 Annotator ..: plyCount ..: 7	Información complementaria de cada posición, Evento, fecha, jugadores, etc. Sólo se presentan las etiquetas con valores significativos, procedentes del fichero fuente PGN.
1. Rxg7 Rxg7 (1... Rh8 2 Kd8 (2... Rf7 3. Qxf7+ Kc ... • Solution	Link con la solución. Un seleccionar este link presentamos u ocultamos el contenido de la solución propuesta.
GPAHS - Generador de Posiciones de Ajedrez en HTML5 Semantico © 2011 UOC	Pié de página con el acrónimo del aplicativo, nombre y copyright de la UOC.

Tabla XLVI: Secciones de la página HTML generada.

3 PRUEBAS Y SIGUIENTES PASOS

Este tercer capítulo de la memoria contiene la evidencia de las pruebas realizadas, así como la presentación de las acciones posteriores que podrían ser realizadas a partir de la entrega del software realizada. Los apartados que componen este capítulo son los siguientes:

Contenido	Apartados	Comentarios
Pruebas	3.1	Pruebas del generador GPAHS.
Siguientes pasos	3.2	Comentarios de posibles acciones de ampliación o mejora sobre la versión de software entregada.

Tabla XLVII: Apartados del capítulo 3.

3.1 Pruebas

En esta apartado de pruebas del generador desarrollado hemos incluido las siguientes secciones:

Contenido	Sección	Comentarios
Entregables	3.1.1	En esta primera sección del apartado se presenta un breve inventario de las carpetas y ficheros que acompañan a la memoria para facilitar las pruebas adicionales que se deseen realizar.
Prueba unitaria	3.1.2	Esta prueba básica consiste en la ejecución del proceso con una única posición contenida en el fichero PGN, pero con todos los pares "etiqueta – valor" cumplimentados. Excepto el campo FEN, todos los campos tienen como valor el nombre de la etiqueta. De esta forma se verifica que se obtiene correctamente todos los valores de cada etiqueta. El campo FEN, por el contrario, sí tiene un valor lógico para poder verificar en la misma prueba que el JavaScript representa correctamente la posición inicial de la partida.
Prueba de verificación	3.1.3	Esta prueba verifica el correcto comportamiento del proceso en el caso de incorrecciones, tanto en la invocación del programa como en el contenido del fichero fuente PGN.
Pruebas integradas	3.1.4	Esta última prueba realiza una generación masiva de posiciones (más de 15.000) para verificar el rendimiento y correcta generación de páginas HTML y bases de conocimiento.
Pruebas Protégé	3.1.5	Utilizando la herramienta Protégé analizaremos si los individuos se han creado correctamente y si contienen toda la información contenida en el fichero PGN de origen.

Tabla XLVIII: Secciones del apartado 3.1.

3.1.1 Entregables

En esta primera sección de este apartado incluimos un inventario de todos los ficheros relacionados con el desarrollo realizado, material que ha sido entregado junto a esta memoria.

Fichero	Carpeta	Contenido
MateInOne.pgn **MateInTwo.pgn** **MateInThree.pgn** **…**	.\pgn\	Ficheros "pgn" que han sido utilizados como entrada de nuestro generador para crear tanto las páginas HTML correspondientes: "MateInOne.html", "MateInTwo.html", "MateInThree.html", etc.; como los ficheros OWL: "MateInOne.owl", "MateInTwo.owl", "MateInThreeOne.owl", etc.
MateInOne.html **MateInTwo.html** **MateInThree.html** **…**	.\html\	Páginas HTML generadas por el programa GPAHS.
style.css	.\html\css\	Hoja de estilos utilizada en las páginas HTML generadas.
chess.gif	.\html\img\	Logo del aplicativo, utilizado en las páginas HTML generadas.
bb.png, bk.png, etc.	.\html\img\30\	Piezas de ajedrez utilizadas en las páginas HTML. Se incluye también otras carpetas con las piezas en otros tamaños.
jquery-1.5.2.js	.\html\js\	Librería que permite crear dinamismo en páginas HTML generadas.
GPAHS.js	.\html\js\	Funciones JQuery de dinamismo, utilizadas en las páginas HTML generadas.
MateInOne.owl **MateInTwo.owl** **MateInThree.owl** **…**	.\owl\	Bases de conocimiento OWL generadas por el programa GPAHS.
CUP$parser$actions.class **Game.class** **lexGPAHS.class** **parser.class** **sym.class** **TablaGames.class**	Raíz	Clases java finales e intermedias generadas por las herramientas utilizadas para generar nuestro programa: "JLex" y "CUP". El programa principal se corresponde con la clase "parser".
cupGPAHS.cup	Raíz	Fichero fuente con la definición del analizador sintáctico, programa principal y métodos auxiliares.
lexGPAHS.lex	Raíz	Fichero fuente con el analizador léxico y clases auxiliares.
CUP_10k.jar	Raíz	Ejecutable CUP.
JLex_1.2.6.jar	Raíz	Ejecutable JLex.
GPAHSDiseñoInicial1.html	Raíz	Página HTML del diseño inicial, incluido en la memoria.
GPAHSDiseñoInicial2.html	Raíz	Página HTML con el segundo diseño, incluido en la memoria.
GPAHSPlantilla.html	Raíz	Plantilla HTML5 estática.
ChessPosition.owl	Raíz	Plantilla OWL estática. Se corresponde con el fichero OWL generado con la herramienta Protégé (integro).
lexGPAHS.java **parser.java** **sym.java**	Raíz	Programas Java generados con las herramientas JLex y CUP.
PFC_compilar.bat	Raíz	Fichero de procesamiento por lotes para compilar el generador.
PFC_ejecutar_integrada.bat **PFC_ejecutar_unitaria.bat** **PFC_ejecutar_verificacion.bat** **…**	Raíz	Ficheros de procesamiento por lotes para ejecutar las distintas pruebas que serán presentadas en este apartado.

Tabla XLIX: Entregables.

3.1.2 Prueba unitaria

Como hemos comentado, en esta sección iniciamos las pruebas con un fichero PGN con una única posición, pero con todas las posibles etiquetas PGN informadas. El fichero PGN tiene el siguiente contenido:

```
[Event "Event"]
[Site "Site"]
[Date "Date"]
[Round "Round"]
[White "White"]
[Black "Black"]
[Result "Result"]
[WhiteTitle "WhiteTitle"]
[BlackTitle "BlackTitle"]
[WhiteElo "WhiteElo"]
[BlackElo "BlackElo"]
[WhiteUSCF "WhiteUSCF"]
[BlackUSCF "BlackUSCF"]
[WhiteNA "WhiteNA"]
[BlackNA "BlackNA"]
[WhiteType "WhiteType"]
[BlackType "BlackType"]
[EventDate "EventDate"]
[EventType "EventType"]
[EventCountry "EventCountry"]
[EventRounds "EventRounds"]
[EventSponsor "EventSponsor"]
[Section "Section"]
[Stage "Stage"]
[Board "Board"]
[Opening "Opening"]
[Variation "Variation"]
[SubVariation "SubVariation"]
[ECO "ECO"]
[NIC "NIC"]
[Time "Time"]
[UTCTime "UTCTime"]
[UTCDate "UTCDate"]
[TimeControl "TimeControl"]
[SetUp "SetUp"]
[FEN "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"]
[Termination "Termination"]
[Annotator "Annotator"]
[Mode "Mode"]
[PlyCount "PlyCount"]
1.move *
```

La prueba realizada ha sido satisfactoria, con los siguientes resultados obtenidos.

Página HTML generada:

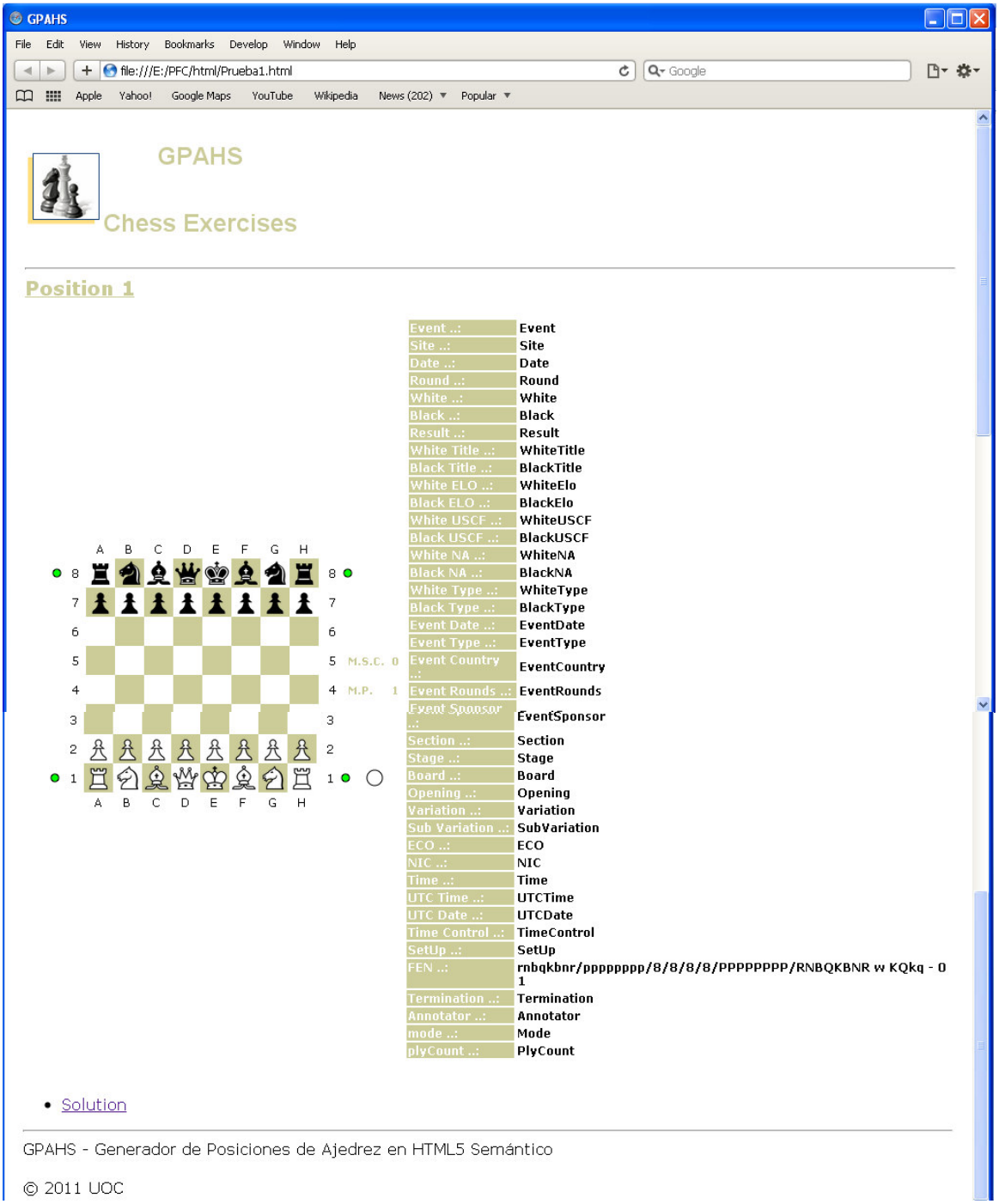


Figura XXX: Página HTML generada.

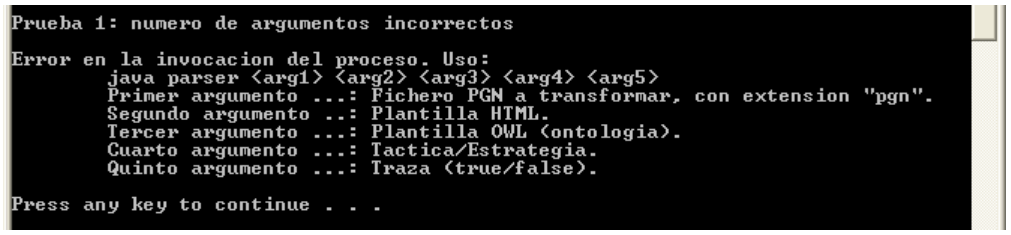
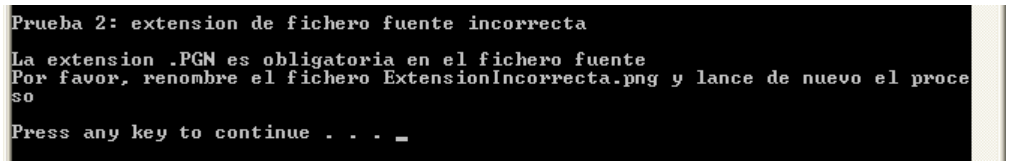
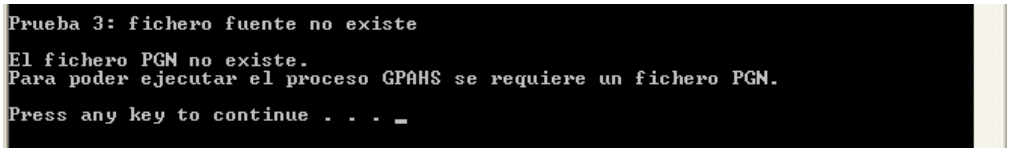
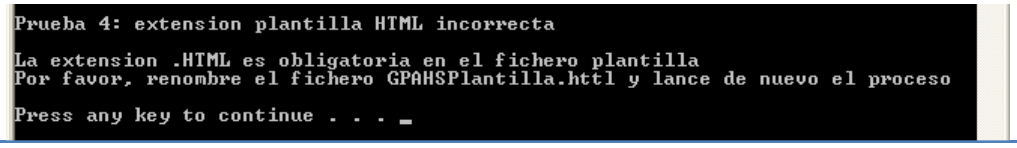
Instancias contenidas en la base de conocimiento generada:

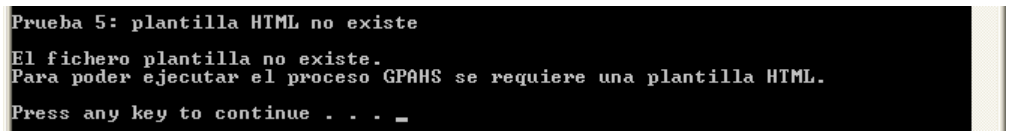
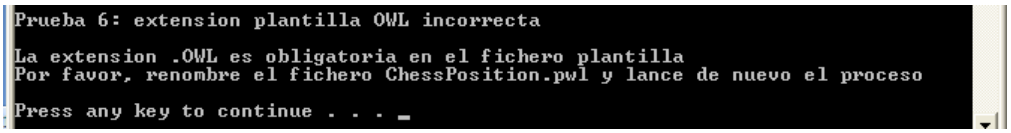
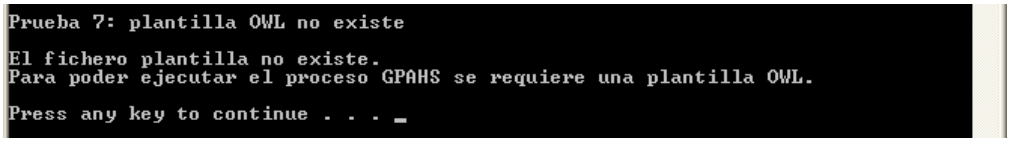
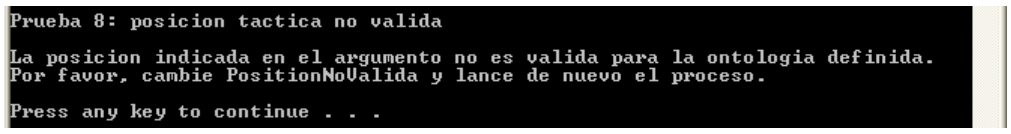
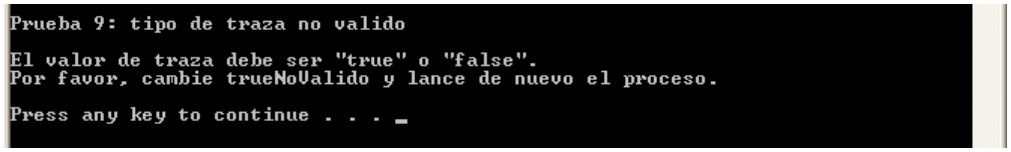
```
...
<!-- Definicion de instancias-->
<Declaration><NamedIndividual IRI="#Event1"/></Declaration>
<ClassAssertion><Class IRI="#Event"/><NamedIndividual IRI="#Event1"/></ClassAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventName"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">Event</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#site"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">Site</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventDate"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">EventDate</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventType"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">EventType</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventCountry"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">EventCountry</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventRounds"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">EventRounds</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eventSponsor"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">EventSponsor</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#section"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">Section</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#stage"/><NamedIndividual IRI="#Event1"/><Literal datatypeIRI="&rdf;PlainLiteral">Stage</Literal></DataPropertyAssertion>
<Declaration><NamedIndividual IRI="#PlayerWhite1"/></Declaration>
<ClassAssertion><Class IRI="#Player"/><NamedIndividual IRI="#PlayerWhite1"/></ClassAssertion>
<Declaration><NamedIndividual IRI="#PlayerBlack1"/></Declaration>
<ClassAssertion><Class IRI="#Player"/><NamedIndividual IRI="#PlayerBlack1"/></ClassAssertion>
<DataPropertyAssertion><DataProperty IRI="#name"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">White</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#title"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">WhiteTitle</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#elo"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">WhiteElo</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#uscf"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">WhiteUSCF</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#na"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">WhiteNA</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#typePlayer"/><NamedIndividual IRI="#PlayerWhite1"/><Literal datatypeIRI="&rdf;PlainLiteral">WhiteType</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#name"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">Black</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#title"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">BlackTitle</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#elo"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">BlackElo</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#uscf"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">BlackUSCF</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#na"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">BlackNA</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#typePlayer"/><NamedIndividual IRI="#PlayerBlack1"/><Literal datatypeIRI="&rdf;PlainLiteral">BlackType</Literal></DataPropertyAssertion>
<Declaration><NamedIndividual IRI="#Position1"/></Declaration>
<ClassAssertion><Class IRI="#Position"/><NamedIndividual IRI="#Position1"/></ClassAssertion>
<DataPropertyAssertion><DataProperty IRI="#fen"/><NamedIndividual IRI="#Position1"/><Literal datatypeIRI="&rdf;PlainLiteral">rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0
1</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#move"/><NamedIndividual IRI="#Position1"/><Literal datatypeIRI="&rdf;PlainLiteral">1.move </Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#typePosition"/><NamedIndividual IRI="#Position1"/><Literal datatypeIRI="&rdf;PlainLiteral">undetermined</Literal></DataPropertyAssertion>
<ObjectPropertyAssertion><ObjectProperty IRI="#hasBoard"/><NamedIndividual IRI="#Position1"/><NamedIndividual IRI="#Board1"/></ObjectPropertyAssertion>
<ObjectPropertyAssertion><ObjectProperty IRI="#hasEvent"/><NamedIndividual IRI="#Position1"/><NamedIndividual IRI="#Event1"/></ObjectPropertyAssertion>
<ObjectPropertyAssertion><ObjectProperty IRI="#hasMatch"/><NamedIndividual IRI="#Position1"/><NamedIndividual IRI="#Match1"/></ObjectPropertyAssertion>
<ObjectPropertyAssertion><ObjectProperty IRI="#hasPlayerWhite"/><NamedIndividual IRI="#Position1"/><NamedIndividual IRI="#PlayerWhite1"/></ObjectPropertyAssertion>
<ObjectPropertyAssertion><ObjectProperty IRI="#hasPlayerBlack"/><NamedIndividual IRI="#Position1"/><NamedIndividual IRI="#PlayerBlack1"/></ObjectPropertyAssertion>
<Declaration><NamedIndividual IRI="#Board1"/></Declaration>
<ClassAssertion><Class IRI="#Board"/><NamedIndividual IRI="#Board1"/></ClassAssertion>
<DataPropertyAssertion><DataProperty IRI="#boardNum"/><NamedIndividual IRI="#Board1"/><Literal datatypeIRI="&rdf;PlainLiteral">Board</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#mode"/><NamedIndividual IRI="#Board1"/><Literal datatypeIRI="&rdf;PlainLiteral">Mode</Literal></DataPropertyAssertion>
<Declaration><NamedIndividual IRI="#Match1"/></Declaration>
<ClassAssertion><Class IRI="#Match"/><NamedIndividual IRI="#Match1"/></ClassAssertion>
<DataPropertyAssertion><DataProperty IRI="#date"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Date</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#round"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Round</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#result"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Result</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#opening"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Opening</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#variation"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Variation</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#subVariation"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">SubVariation</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#eco"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">ECO</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#nic"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">NIC</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#time"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Time</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#utcTime"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">UTCTime</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#utcDate"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">UTCDate</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#timeControl"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">TimeControl</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#setUp"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">SetUp</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#termination"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Termination</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#annotator"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">Annotator</Literal></DataPropertyAssertion>
<DataPropertyAssertion><DataProperty IRI="#plyCount"/><NamedIndividual IRI="#Match1"/><Literal datatypeIRI="&rdf;PlainLiteral">PlyCount</Literal></DataPropertyAssertion>
</Ontology>
<!-- Generated by GPAHS-->
```

En las evidencias anteriores podemos verificar que el generador se ha comportado como esterábamos, se ha creado una posición HTML donde se ha incluido toda la información contenida en el fichero origen, y se ha generado una base de conocimiento con una instancia que contiene igualmente toda la información contenida en el fichero origen.

3.1.3 Prueba de verificación

En esta sección analizaremos el comportamiento del proceso en el caso de encontrar errores en la invocación o en la verificación de contenido PGN. En concreto, las pruebas a realizar son las siguientes:

Prueba	Descripción	Resultado
1	El proceso debe verificar el número correcto de argumentos. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
2	El fichero fuente debe tener la extensión “pgn”. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
3	El primer parámetro debe hacer referencia a un fichero PGN que exista en la subcarpeta “pgn”. La invocación se debe realizar sin el nombre de la subcarpeta. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
4	El fichero que será utilizado como plantilla HTML debe tener la extensión “html”. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK

5	El fichero que será utilizado como plantilla HTML debe existir en la carpeta desde donde se está invocando al proceso “parser” (generador). En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
6	El fichero que será utilizado como plantilla OWL debe tener la extensión “owl”. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
7	El fichero que será utilizado como plantilla OWL debe existir en la carpeta desde donde se está invocando al proceso “parser” (generador). En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
8	La posición táctica/estratégica debe ser válida, en el sentido de que debe ser una de las clases definidas en la ontología. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK
9	El indicador de traza sólo acepta los valores “true” o “false”. En caso contrario, se debe mostrar un mensaje por consola y cancelar la ejecución del proceso. Evidencia de la prueba realizada: 	OK

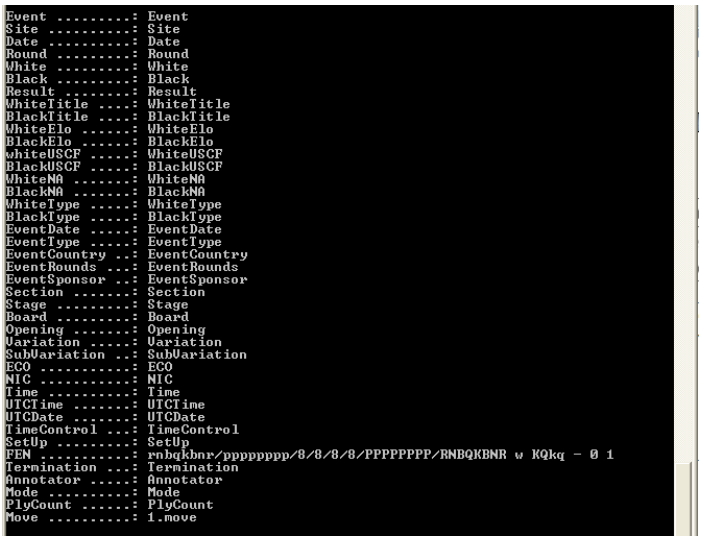
10	En esta prueba se verifica que tras la invocación del proceso con activación de traza (último argumento con valor “true”), los mensajes de consola son mostrados correctamente. Evidencia de la prueba realizada:  	OK

Tabla L: Prueba de verificación.

3.1.4 Pruebas integradas

En esta sección mostramos el resumen de los ficheros PGN utilizados como entrada de nuestro proceso, y hacemos un breve resumen de las principales fuentes de información utilizadas para la elaboración de estos ficheros.

Los ficheros PGN tratados han sido los siguientes:

Fichero	Táctica / Estategia asociada	Posiciones
200CombinacionesDeMatePolgar.pgn	Combination	200
1001Reinfeld.pgn	Combination	1.001
AnthologyofChessCombinations3.pgn	Combination	2.709
MateInOne.pgn	MateInOne	306
MateInTwo.pgn	MateInTwo	3.411
MateInThree.pgn	MateInThree	744
Otherendgame.pgn	OtherEndgame	144
tactics.pgn	Combination	928
TacticalAuerswald1.pgn	PositionMiddle	1.725
TacticalAuerswald2.pgn	PositionMiddle	1.792
Tactica1.pgn	PositionMiddle	100
Tactica2.pgn	PositionMiddle	100
Aperturas.pgn	PositionBegin	2.013
		15.173

Tabla LI: Ficheros PGN procesados.

Como puede verse en los ficheros que acompañan a esta memoria, más de 15.000 posiciones han sido generadas satisfactoriamente a partir de la información contenida en los ficheros PGN anteriores.

Incluimos a continuación una breve reseña de las fuentes de información de ficheros PGN más relevantes.

Combinaciones de Judit Polgár

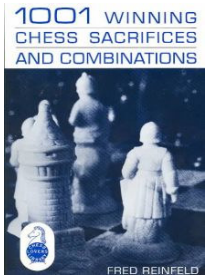
Judit Polgár, nacida el 23 de julio de 1976 en Hungría, es considerada por muchos la mejor jugadora de ajedrez de la historia. Posee el título de *Gran Maestro Internacional*. En octubre de 2008 ocupaba la posición vigésima séptima del mundo según la clasificación de la FIDE (que incluye hombres y mujeres) con una puntuación ELO de 2711. Es la única mujer que ha conseguido estar entre los diez primeros ajedrecistas de la clasificación mundial, lográndolo en la lista de enero de 1996. En 1998 consiguió la medalla de oro en la Olimpiada de Ajedrez de Salónica (Grecia).

Figura XXXI: Foto de Judit Polgár.

Hay mucha documentación desarrollada por esta jugadora, dado que gran parte de su tiempo lo dedica a la preparación y enseñanza de otros jugadores. Para esta prueba hemos seleccionado una conocida base de datos de 200 posiciones de combinaciones, creada por esta jugadora, de gran interés por su gran valor didáctico.

Fichero fuente: "200CombinacionesDeMatePolgar.pgn"

Fred Reinfeld

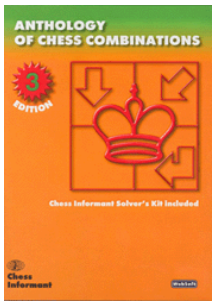


Fred Reinfeld (1910 – 1964) fue uno de los maestros de ajedrez norteamericano con mayor número de publicaciones de calidad editadas. De sus más de 100 obras, hemos utilizado como fuente PGN una de las más conocidas, "*1001 Brilliant Chess Sacrifices and Combinations*", editada en 1955 y pasada a formato PGN hace pocos años. Fichero: "1001Reinfeld.pgn"

Figura XXXII: Libro de Fred Reinfeld.

NOTA: *Este fichero ha sido incluido porque los valores de las etiquetas han permitido validar y depurar el analizador léxico/sintáctico presentado, pero como se puede ver la página HTML generada, no siempre se presentan las soluciones de cada posición. Para obtenerlas, se requiere la adquisición del libro.*

Antología de las combinaciones 3ª edición



La base de datos PGN "*Antología de las combinaciones 3ra edición*", de la casa Chess Informant, es una colección de 2709 combinaciones, las primeras 300 son instructivas, y están clasificadas con la codificación de combinaciones de chess informant:

- I: Significa combinaciones con ataque de mate,
- II: Significa combinaciones para buscar tablas, y
- III: Significa otras combinaciones.

Figura XXXIII: Libro de Chess Informant.

Las letras que acompañan a la clasificación anterior significan lo siguiente:

- "a": Ataque a la descubierta / "Aniquilación de la defensa",
- "b": Bloqueo,
- "c": Despeje de espacio,
- "d" Desviación,
- "e" Ataque a la descubierta,
- "f": Clavada.
- "g": Destrucción de la estructura de peones,

- "h": Encaminamiento,
- "i" Intercepción y/o Interferencia,
- "j": Ataque Doble.

Estos datos aparecen al inicio de la etiqueta “event”. Además, en la etiqueta “date” se indica el nivel de dificultad, con el valor ELO equivalente. Fichero fuente: “AnthologyofChessCombinations3.PGN”.

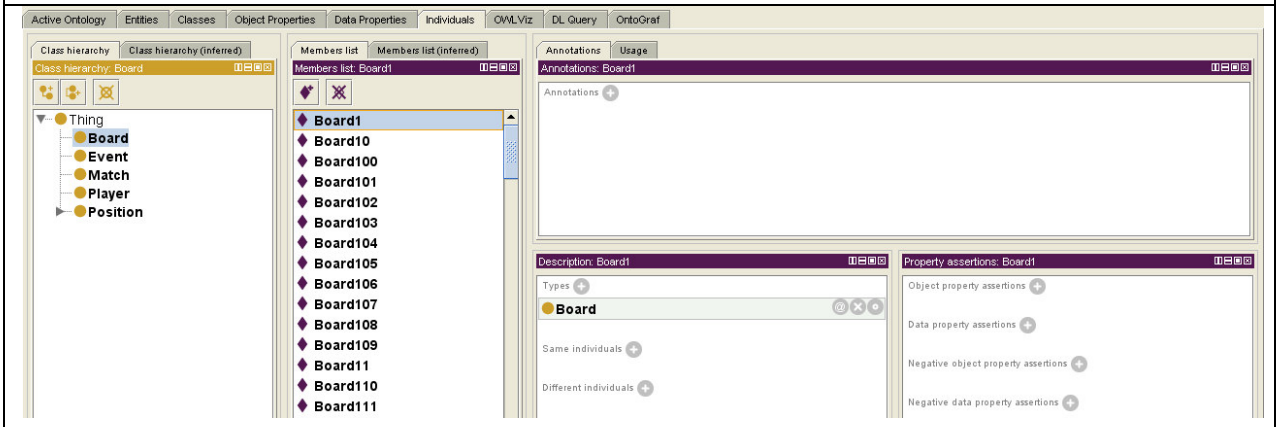
3.1.5 Pruebas Protégé

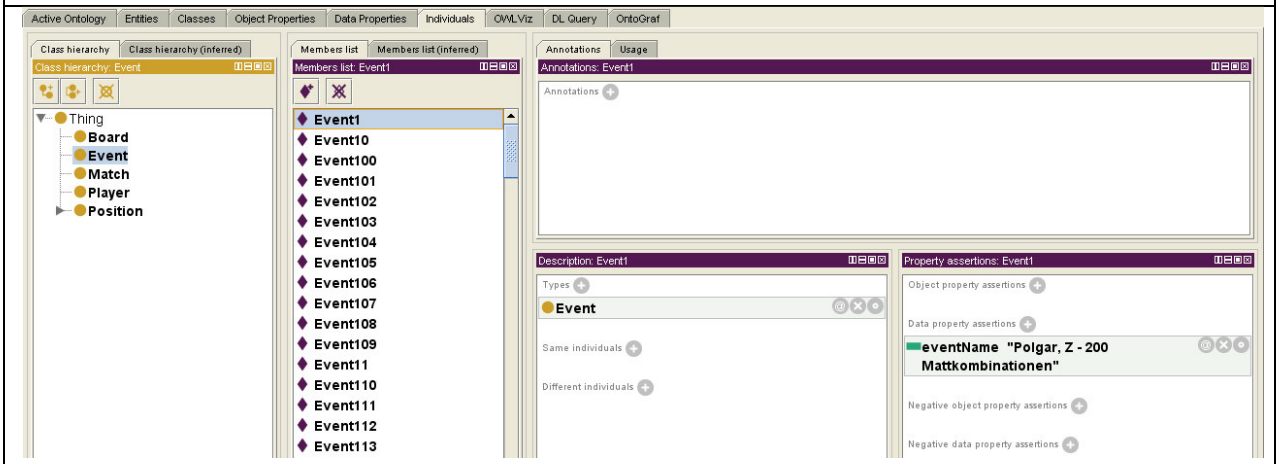
En esta última sección mostramos la evidencia de que los individuos incluidos en las bases de conocimiento generadas se han creado correctamente. Para ello, tomaremos como referencia la primera posición de cada fichero de entrada y mostraremos su contenido con la ayuda de Protégé. En primer lugar mostramos el contenido de una posición en formato PGN de cada uno de los ficheros utilizados en las pruenas y a continuación se muestra como se visualiza dicha información en el editor Protégé.

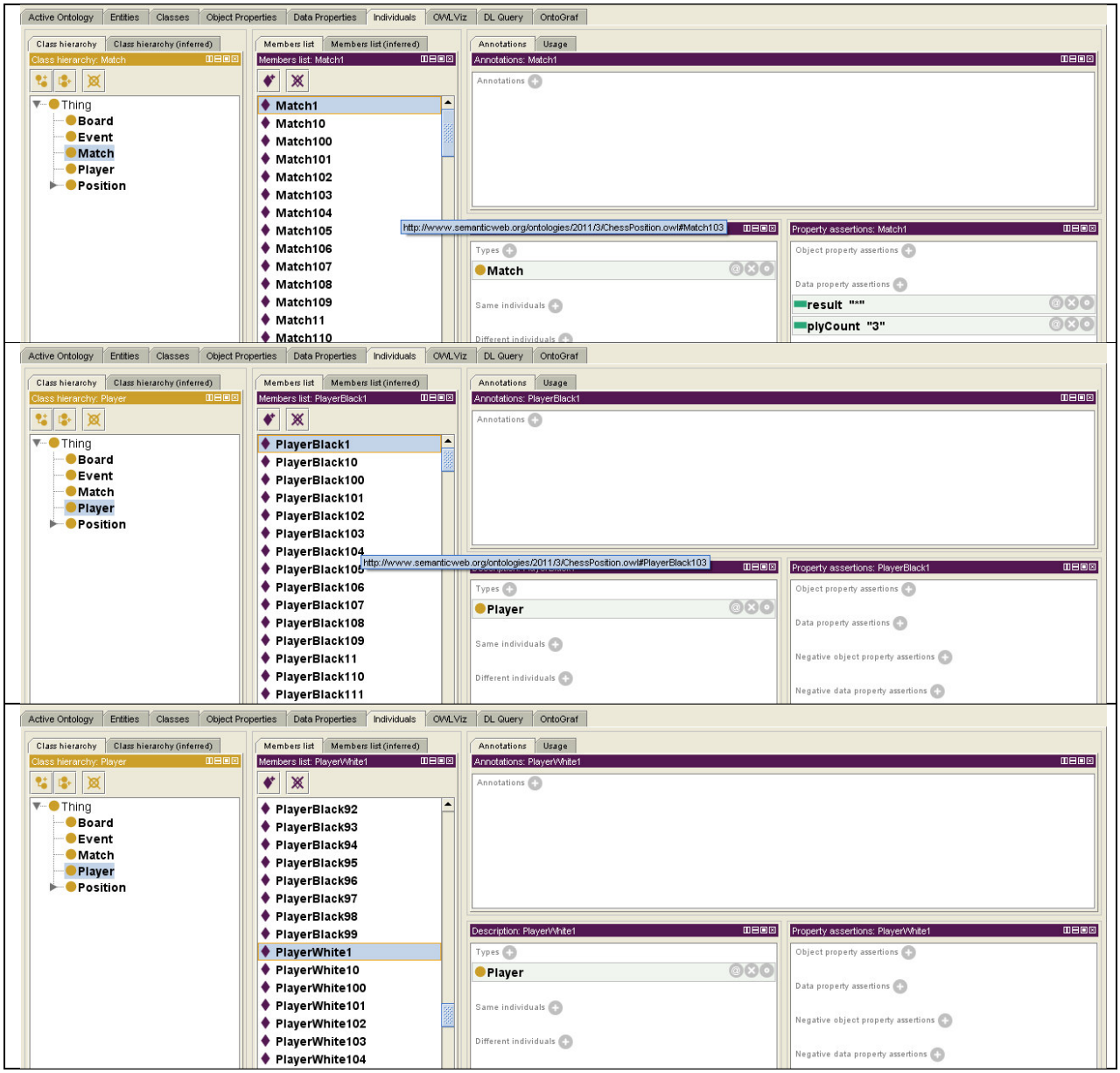
Fichero "200CombinacionesDeMatePolgar.pgn":

[Event "Polgar, Z - 200 Mattkombinationen"]
[Site "?"]
[Date "????.??.??"]
[Round "?"]
[White "?"]
[Black "?"]
[Result "*"]
[SetUp "1"]
[FEN "r1bqk2r/pppn1ppp/5n2/2B5/2B1p3/2N5/PPP1Q1PP/R4RK1 w kq - 0 1"]
[PlyCount "3"]

1. Qxe4+ Nxe4 2. Bxf7# *






Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Match
Members list
Members list (inferred)
Members list: Match1
Annotations
Usage
Annotations: Match1
Thing
Board
Event
Match
Player
Position
Match1
Match10
Match100
Match101
Match102
Match103
Match104
Match105
Match106
Match107
Match108
Match109
Match11
Match110
http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl#Match103
Property assertions: Match1
Types
Match
Same individuals
Different individuals
Object property assertions
Data property assertions
result ""
plyCount "3"
Class hierarchy: Player
Members list: PlayerBlack1
Annotations: PlayerBlack1
PlayerBlack1
PlayerBlack10
PlayerBlack100
PlayerBlack101
PlayerBlack102
PlayerBlack103
PlayerBlack104
PlayerBlack105
PlayerBlack106
PlayerBlack107
PlayerBlack108
PlayerBlack109
PlayerBlack11
PlayerBlack110
PlayerBlack111
http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl#PlayerBlack103
Property assertions: PlayerBlack1
Player
Negative object property assertions
Negative data property assertions
Members list: PlayerWhite1
Annotations: PlayerWhite1
PlayerBlack92
PlayerBlack93
PlayerBlack94
PlayerBlack95
PlayerBlack96
PlayerBlack97
PlayerBlack98
PlayerBlack99
PlayerWhite1
PlayerWhite10
PlayerWhite100
PlayerWhite101
PlayerWhite102
PlayerWhite103
PlayerWhite104
Description: PlayerWhite1
Property assertions: PlayerWhite1

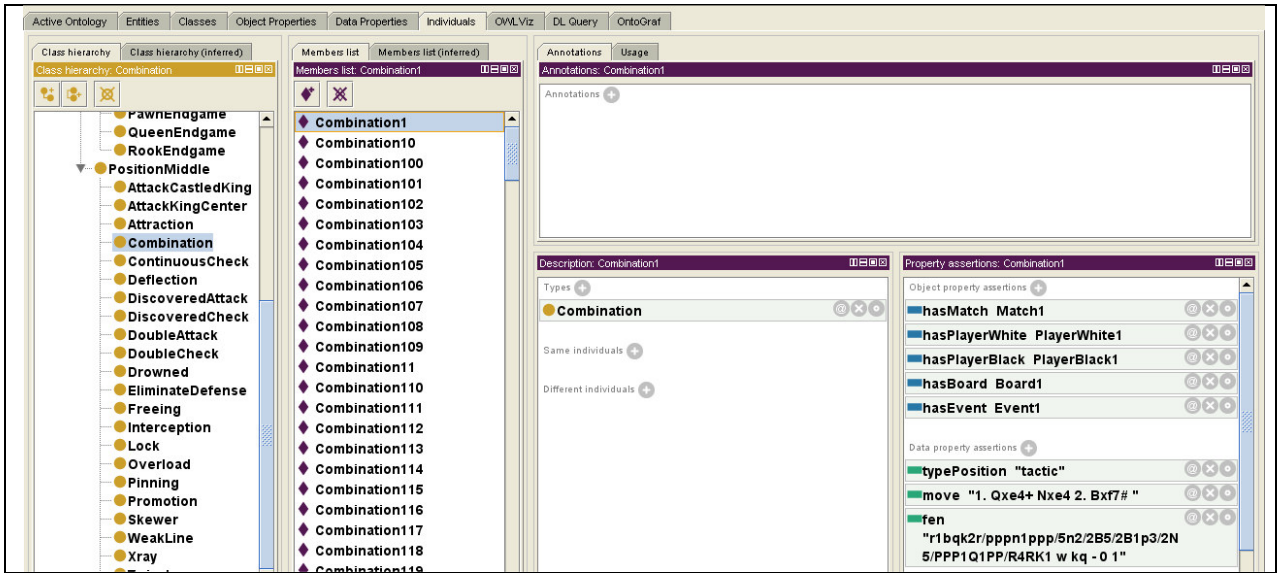
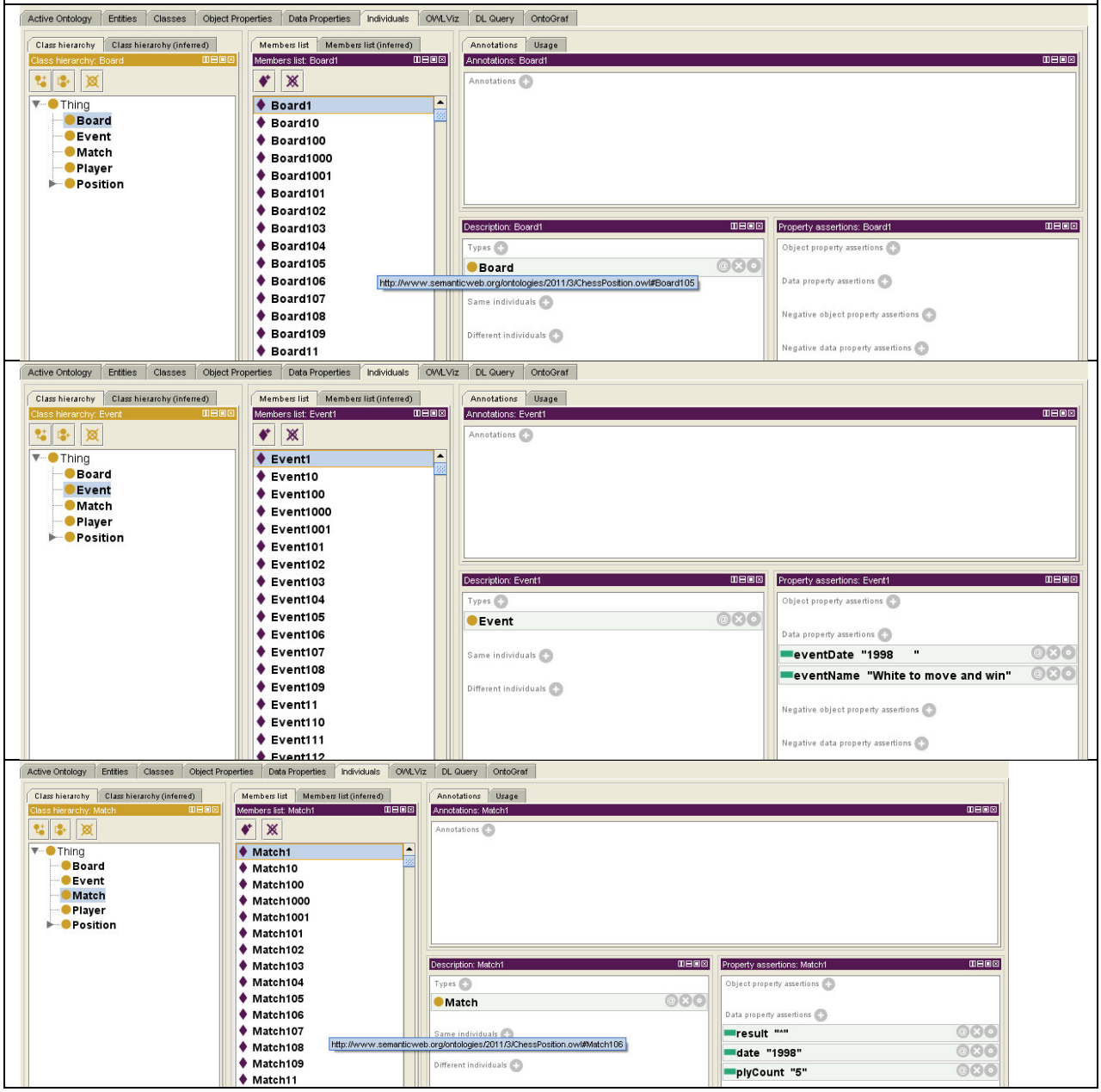


Tabla LII: Pruebas Protégé (I).

Fichero "1001Reinfeld.pgn":

[Event "White to move & win"]
[Site "?"]
[Date "1998.??.??"]
[Round "?"]
[White "1001 Winning Chess Sacrifices"]
[Black "and Combinations"]
[Result "*"]
[SetUp "1"]
[FEN "3r2k1/1p5p/6p1/p2q1p2/P1Q5/1P5P/1P6/5RK1 w - - 0 1"]
[PlyCount "5"]
[EventDate "1998.??.??"]

1. Rd1 Qxc4 2. Rxd8+ Kf7 3. bxc4 *



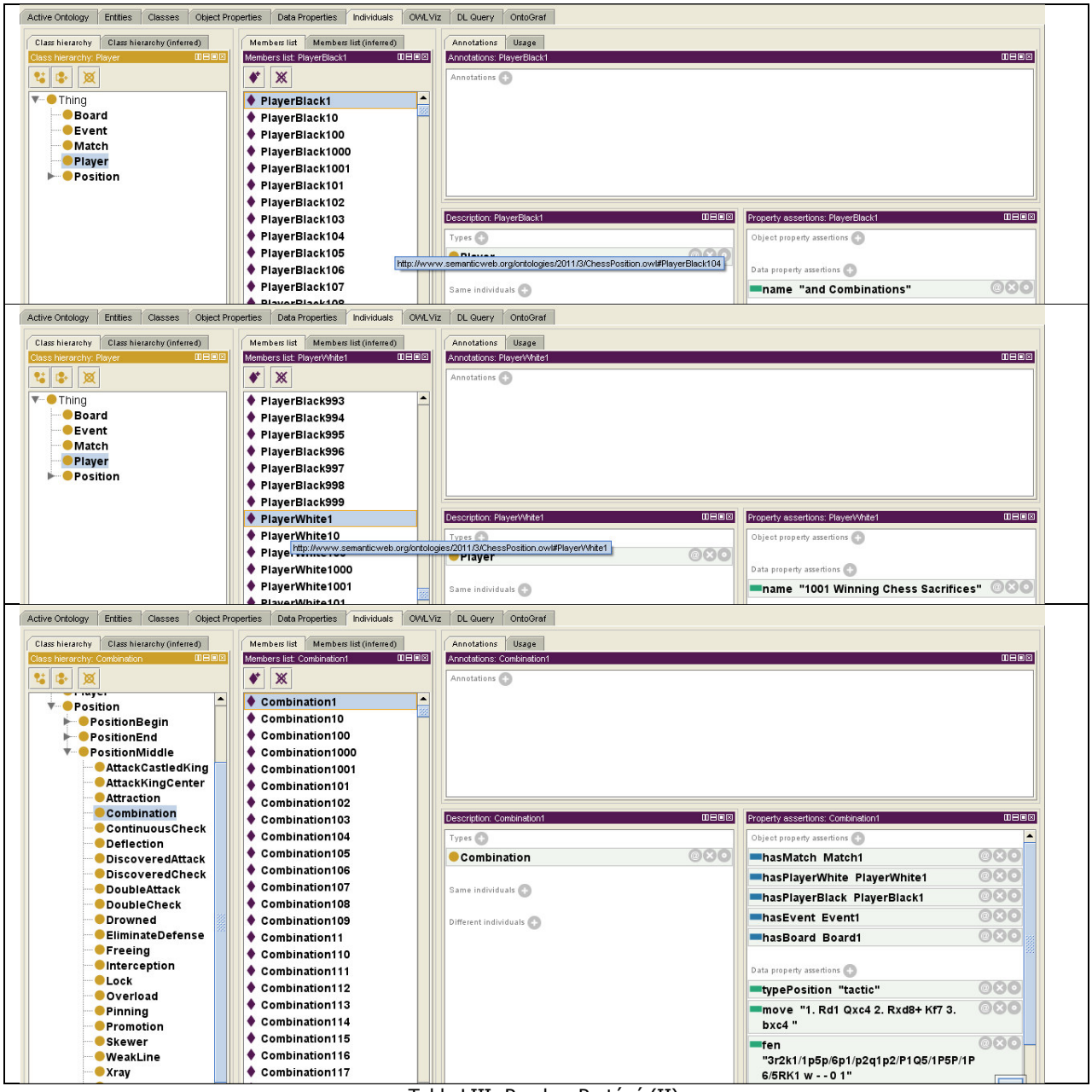
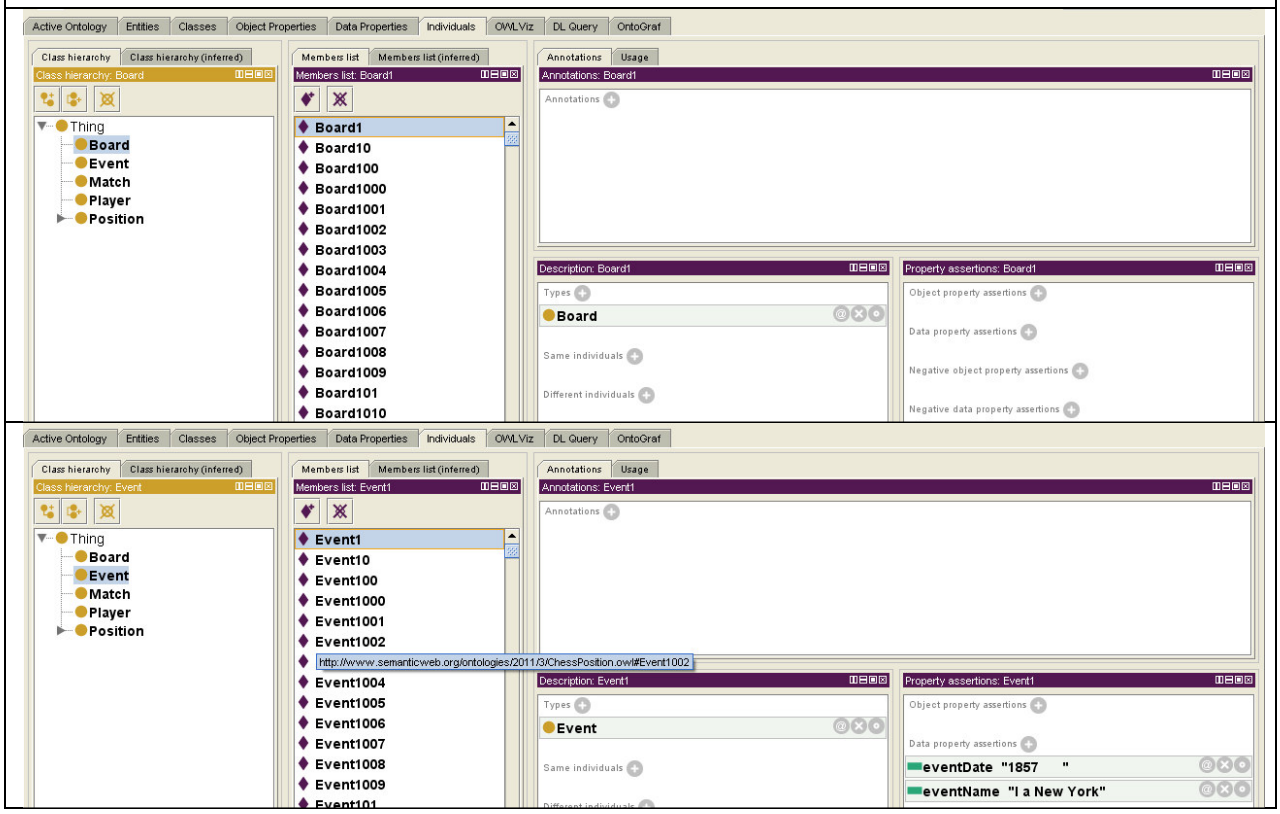


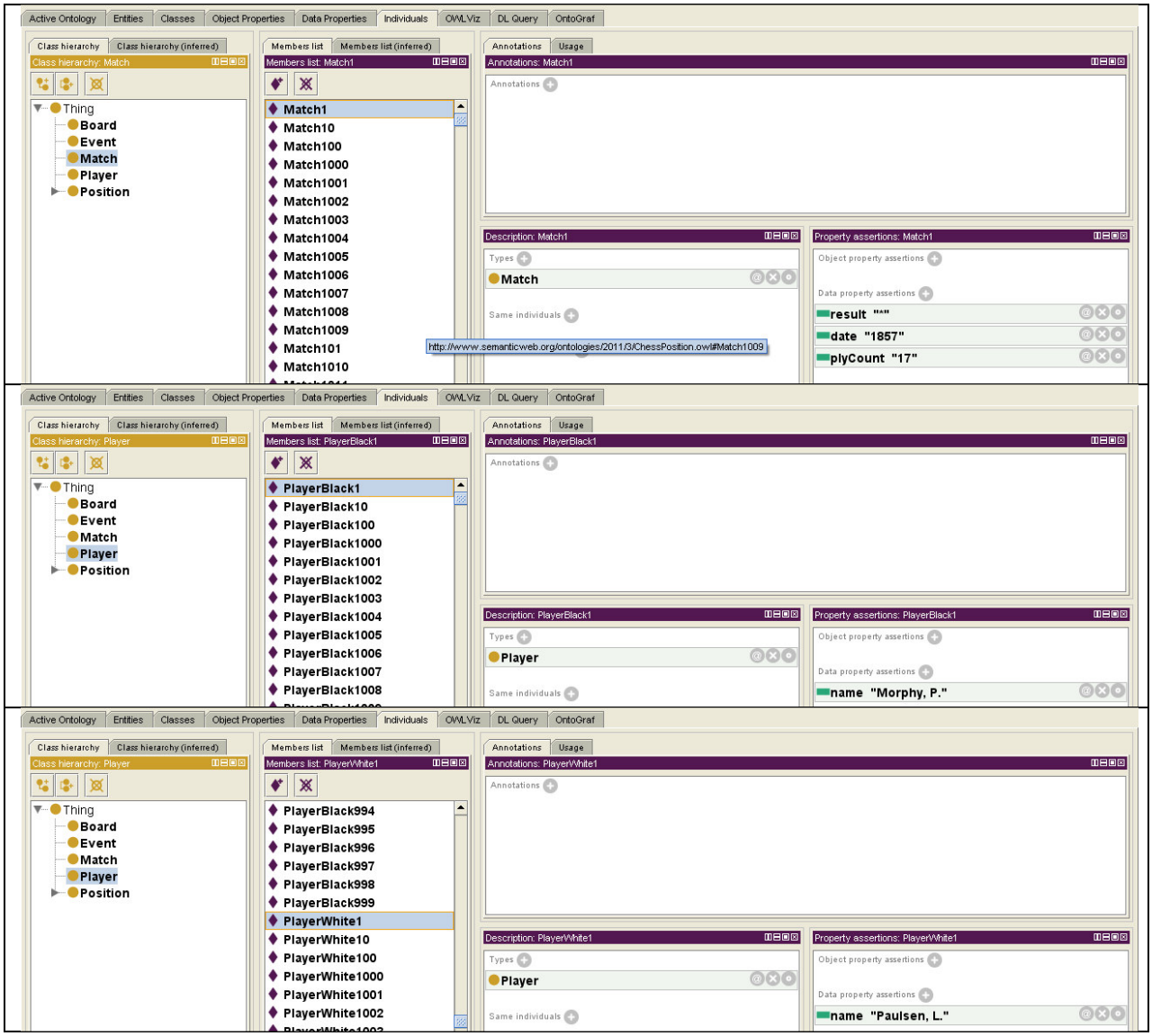
Tabla LIII: Pruebas Protégé (II).

Fichero "AnthologyofChessCombinations3.pgn":

```
[Event "I a New York"]
[Site "?"]
[Date "1857.??.??"]
[Round "?"]
[White "Paulsen, L."]
[Black "Morphy, P."]
[Result "*"]
[SetUp "1"]
[FEN "4r1k1/p1pb1ppp/Qbp1r3/8/1P6/2Pq1B2/R2P1PPP/2B2RK1 b - - 0 1"]
[PlyCount "17"]
[EventDate "1857.??.??"]

1... Qxf3 $1 2. gxf3 Rg6+ 3. Kh1 Bh3 4. Rd1 (4. Qd3 f5 5. Qc4+ 5... Kf8 $1 6.
Qh4 (6. Qf4 6... Bxf2 $1 $19) 6... Bxf1 7. h3 Bg2+ 8. Kh2 8... Bxf3 $19) 4...
Bg2+ 5. Kg1 Bxf3+ 6. Kf1 6... Rg2 $1 (6... Bg2+ 7. Kg1 7... Be4+ $1 (7... Bh3+
8. Kh1 Bxf2 9. Qf1 Bxf1 10. Rxf1 10... Re2 $19 {Paulsen - Morphy}) 8. Kf1 8...
Bf5 $1 9. Qe2 Bh3+ 10. Ke1 10... Rg1# {#}) 7. Qd3 (7. Qxb6 7... Rxh2 {
#C5 #CBh1#}) 7... Rxf2+ 8. Kg1 Rg2+ 9. Kh1 (9. Kf1 9... Rg1# {#}) 9... Rg1# {#}
*
```




Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Match
Thing
Board
Event
Match
Player
Position
Members list
Members list (inferred)
Members list: Match1
Match1
Match10
Match100
Match1000
Match1001
Match1002
Match1003
Match1004
Match1005
Match1006
Match1007
Match1008
Match1009
Match101
Match1010
Annotations
Usage
Annotations: Match1
Description: Match1
Types
Match
Same individuals
Property assertions: Match1
Object property assertions
Data property assertions
result ""
date "1857"
plyCount "17"
http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl#Match1009
Class hierarchy: Player
Members list: PlayerBlack1
PlayerBlack1
PlayerBlack10
PlayerBlack100
PlayerBlack1000
PlayerBlack1001
PlayerBlack1002
PlayerBlack1003
PlayerBlack1004
PlayerBlack1005
PlayerBlack1006
PlayerBlack1007
PlayerBlack1008
Annotations: PlayerBlack1
Description: PlayerBlack1
Player
Property assertions: PlayerBlack1
name "Morphy, P."
Members list: PlayerWhite1
PlayerBlack994
PlayerBlack995
PlayerBlack996
PlayerBlack997
PlayerBlack998
PlayerBlack999
PlayerWhite1
PlayerWhite10
PlayerWhite100
PlayerWhite1000
PlayerWhite1001
PlayerWhite1002
Annotations: PlayerWhite1
Description: PlayerWhite1
Property assertions: PlayerWhite1
name "Paulsen, L."

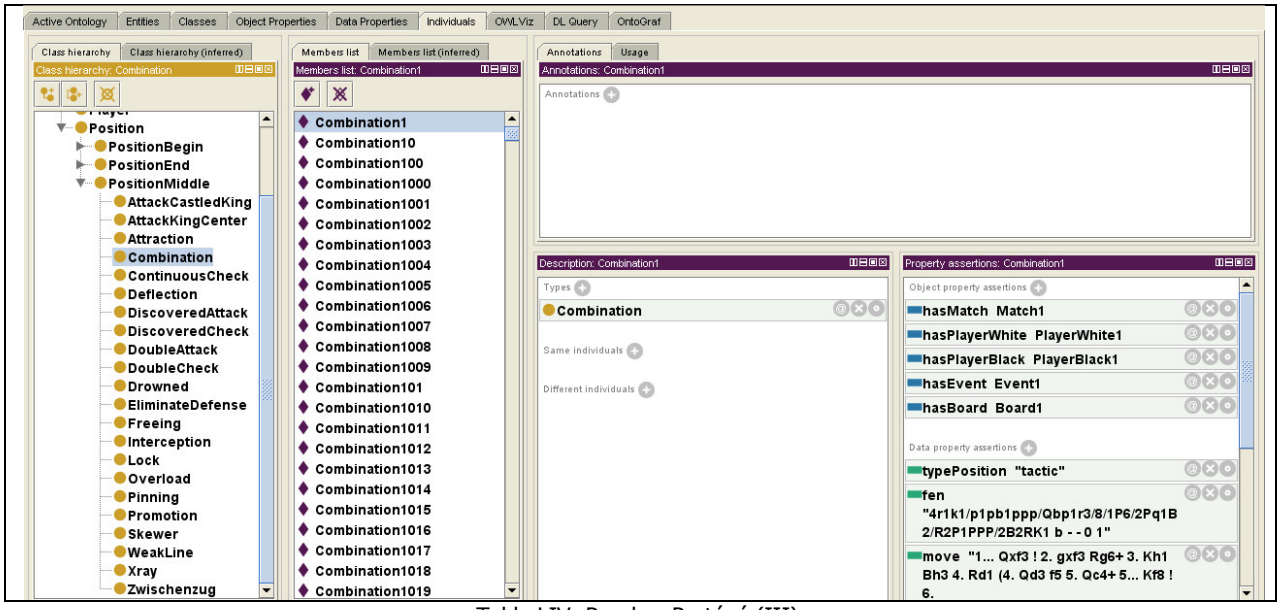
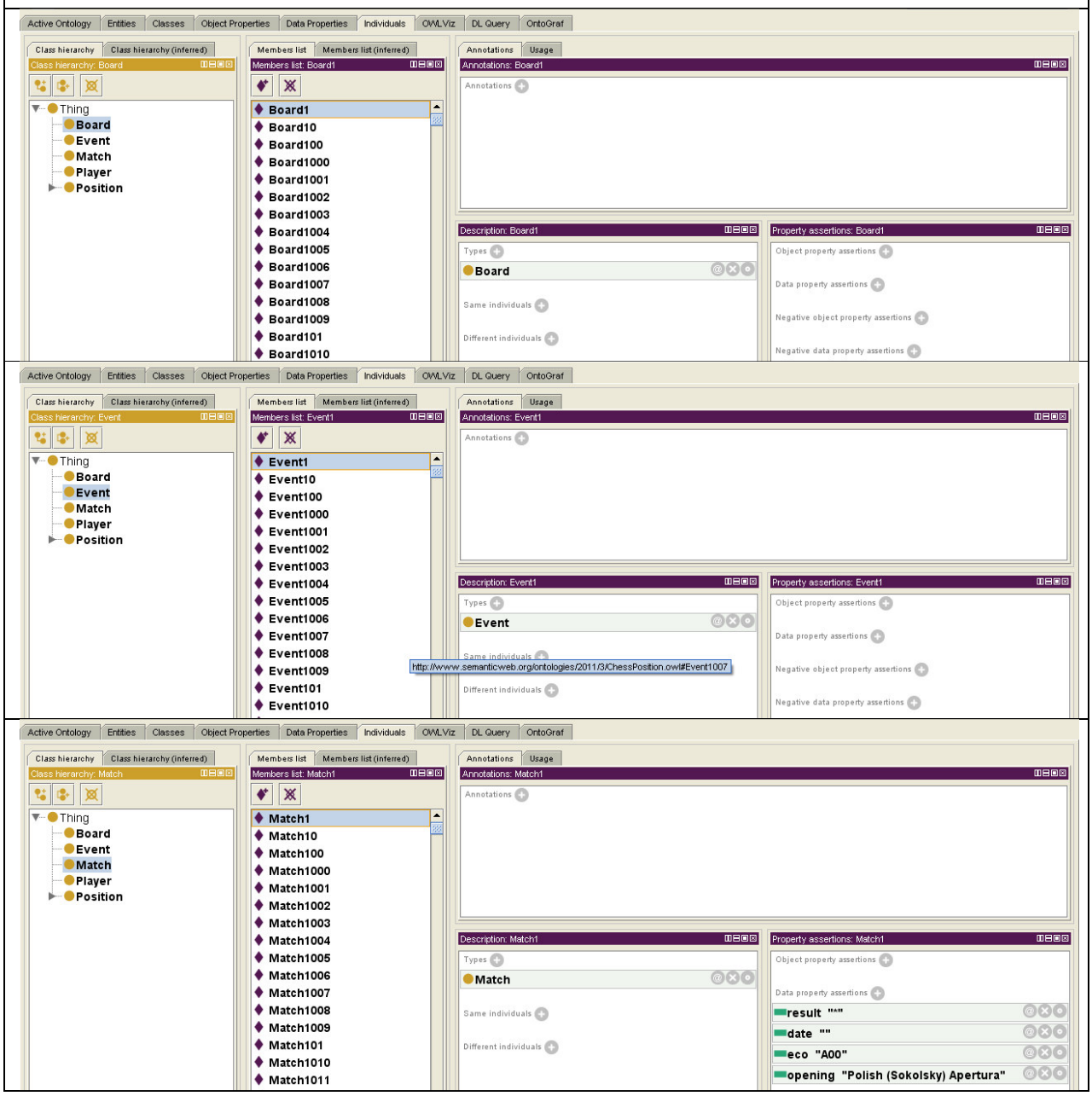


Tabla LIV: Pruebas Protégé (III).

Fichero "Aperturas.pgn":

[Event "?"]
[Site "?"]
[Date "??.??.??"]
[White "?"]
[Black "?"]
[Result "*"]
[ECO "A00"]
[Opening "Polish (Sokolsky) Apertura"]

1.b4 *



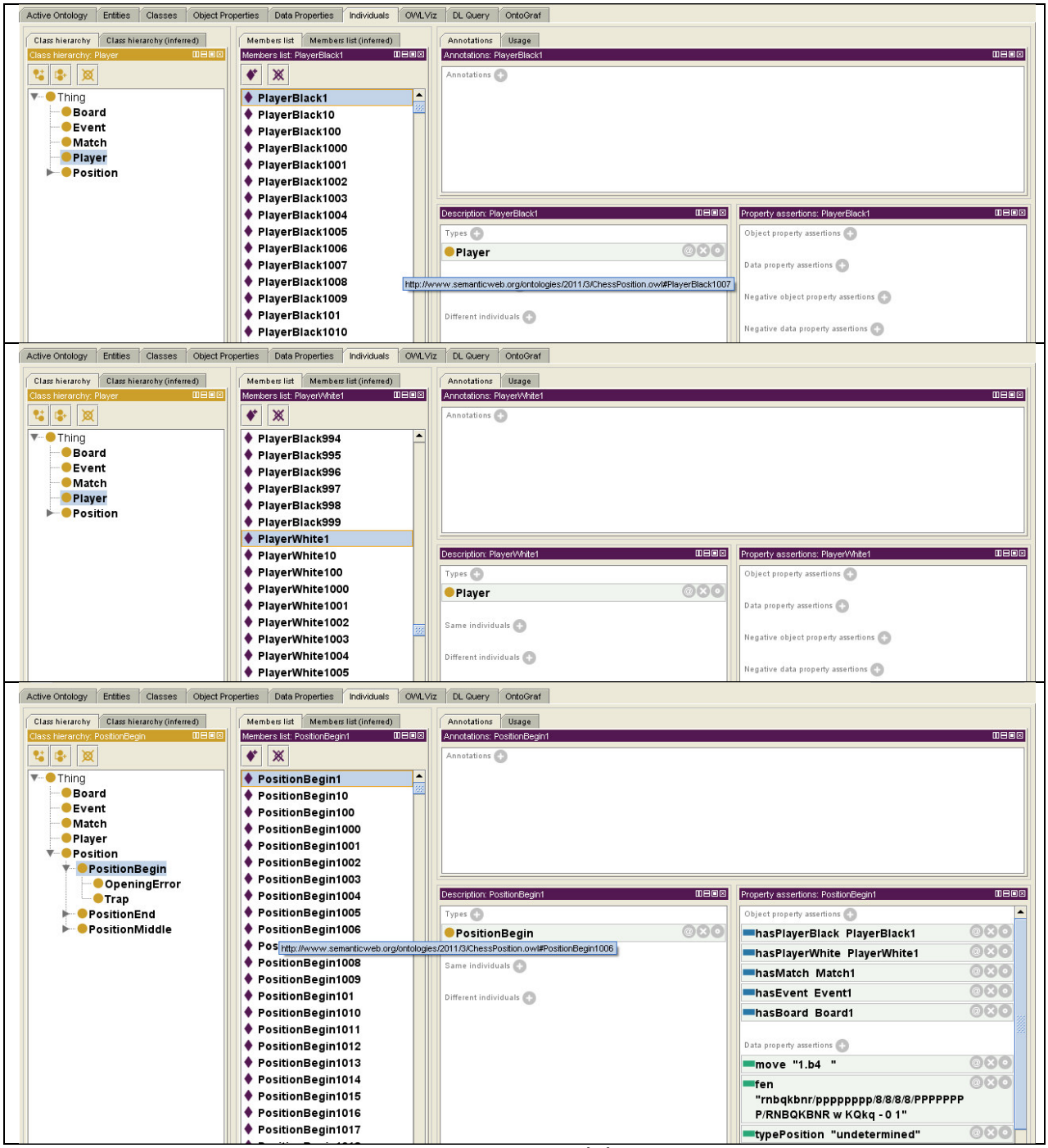
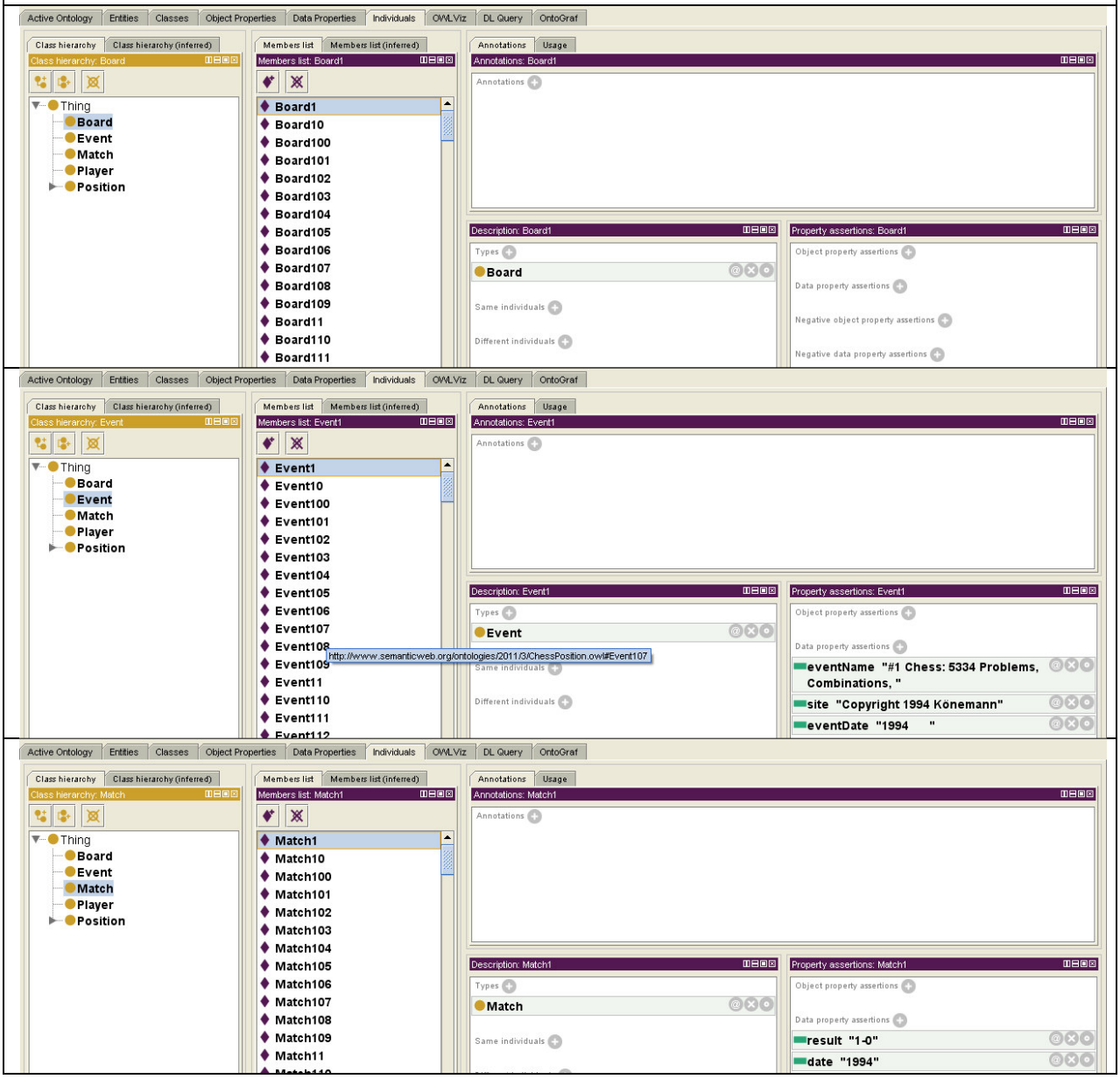


Tabla LV: Pruebas Protégé (IV).

Fichero "MateInOne.pgn":
[Event "#1 Chess: 5334 Problems, Combinations, "]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in one"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "3q1rk1/5pbp/5Qp1/8/8/2B5/5PPP/6K1 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Qxg7# 1-0



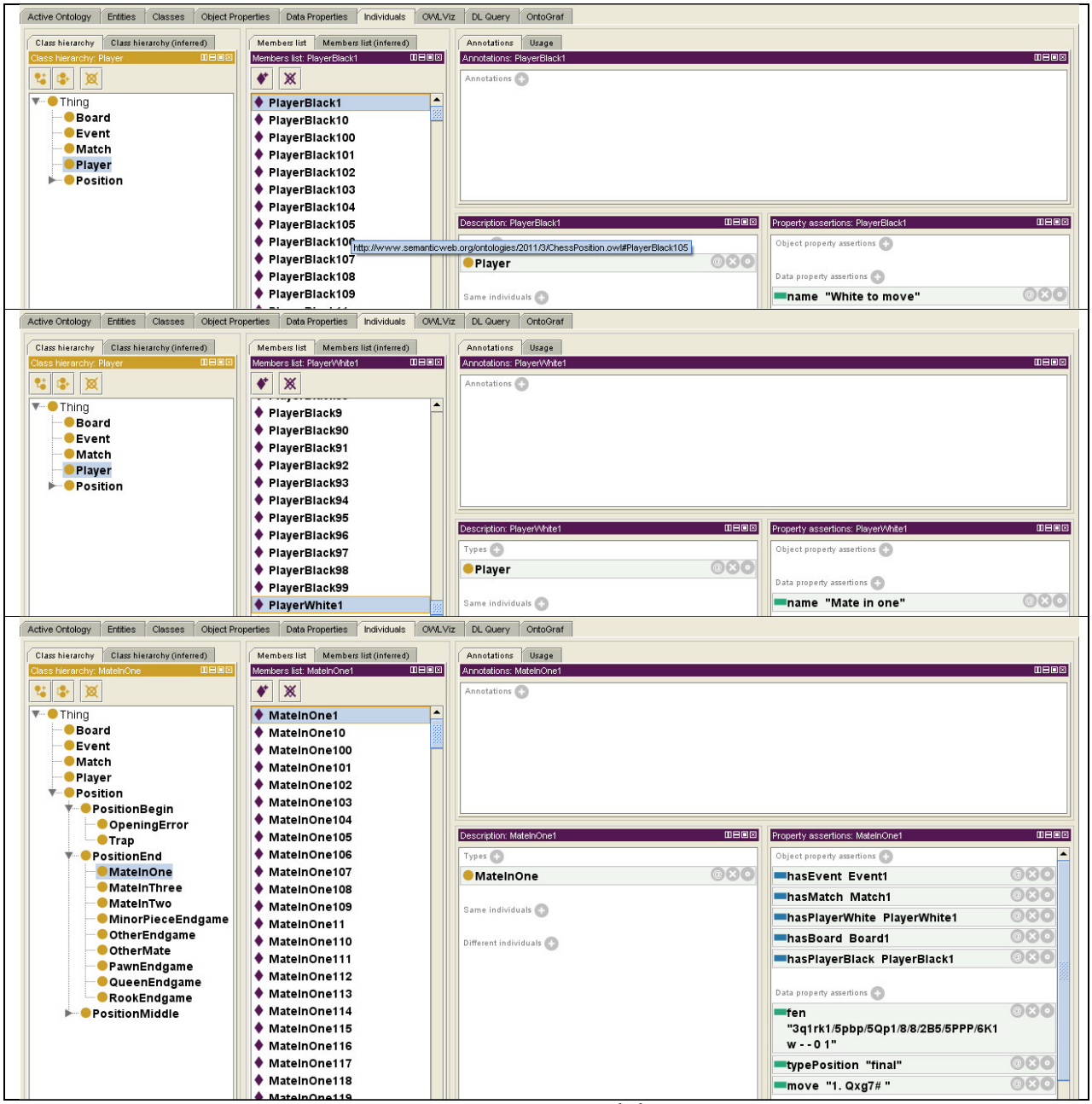
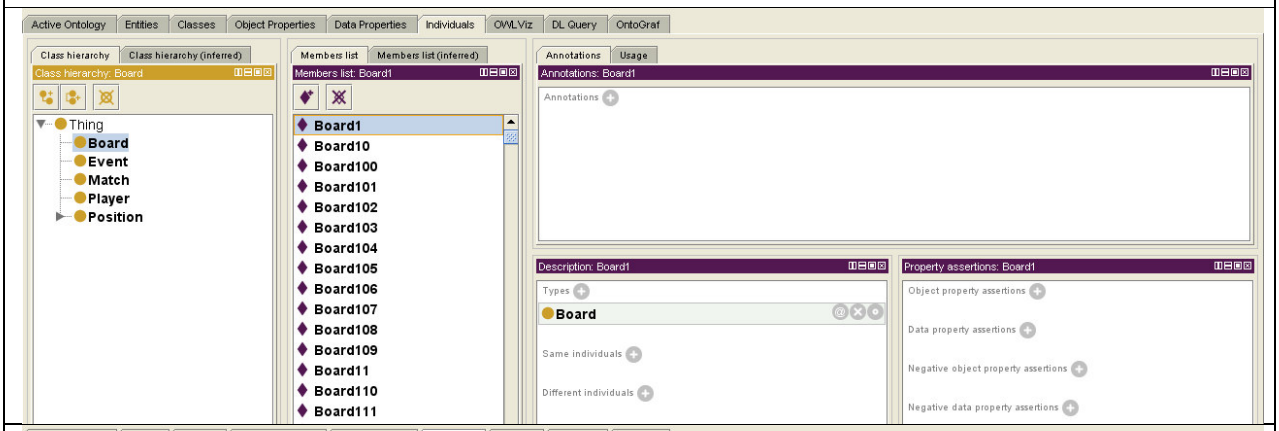


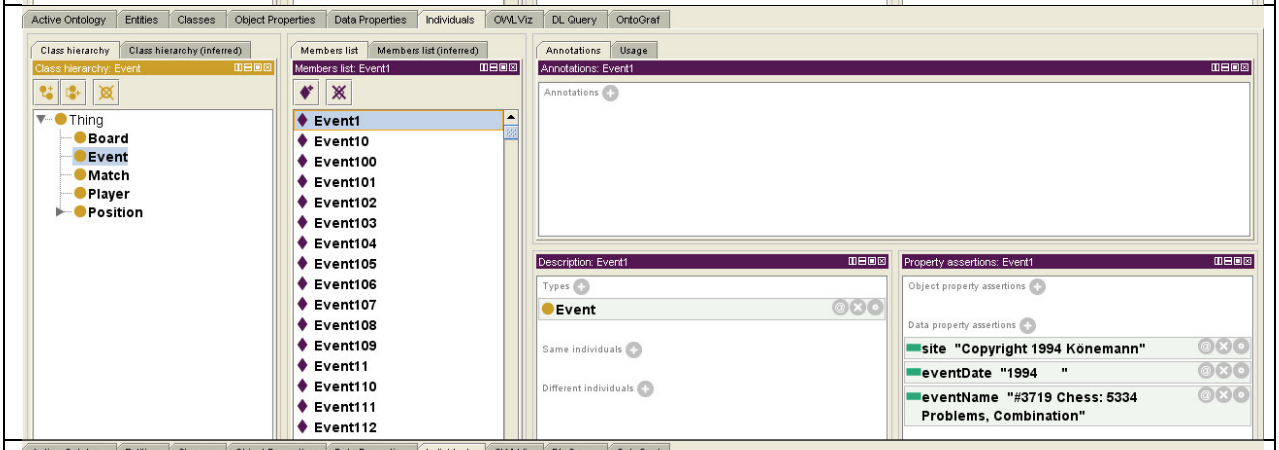
Tabla LVI: Pruebas Protégé (V).

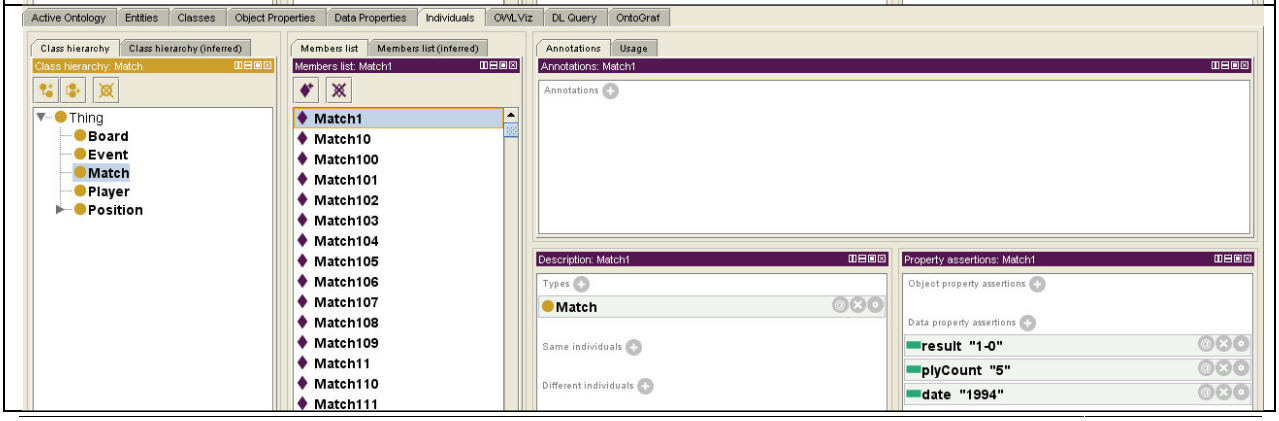
Fichero "MateInThree.pgn":

[Event "#3719 Chess: 5334 Problems, Combination"]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in three"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "4rr2/2p1n1R1/pq1pkp2/1N6/BpppN1P1/b5B1/8/3K3Q w - - 0 1"]
[PlyCount "5"]
[EventDate "1994.??.??"]

1. Nc5+ Qxc5 (1... dxc5 2. Nxd4+ cxd4 3. Qe4#) 2. Nxc7+ Qxc7 3. Qe4# 1-0







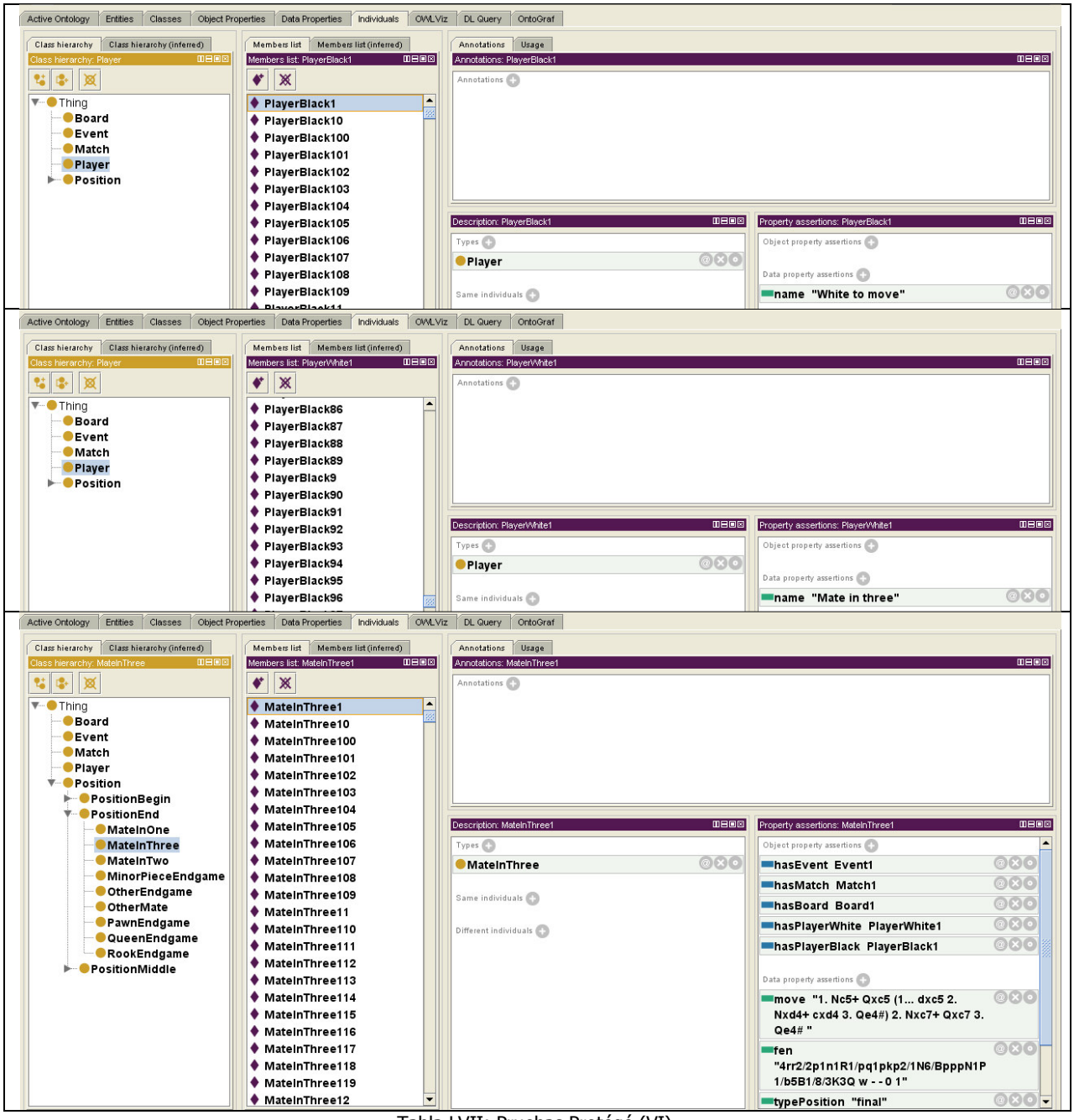
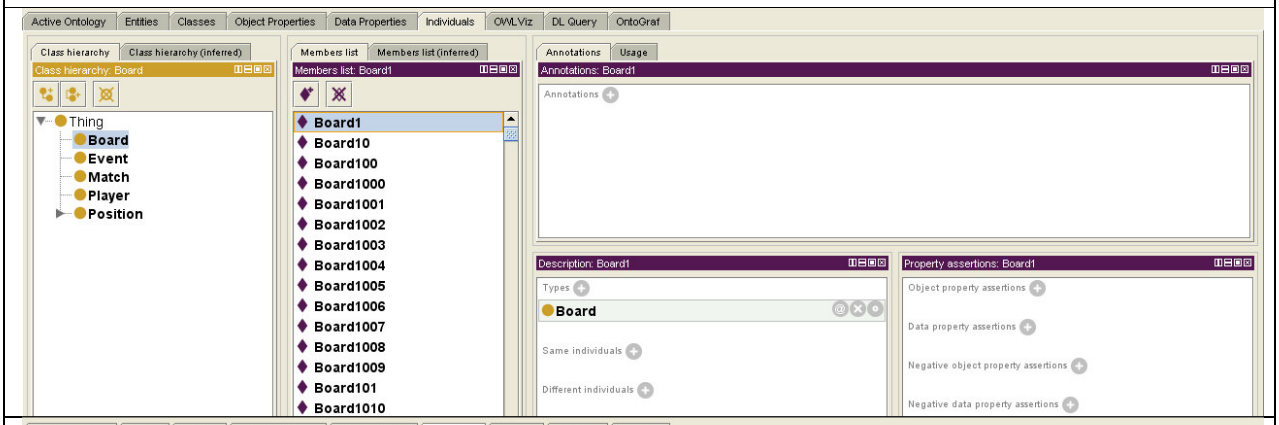


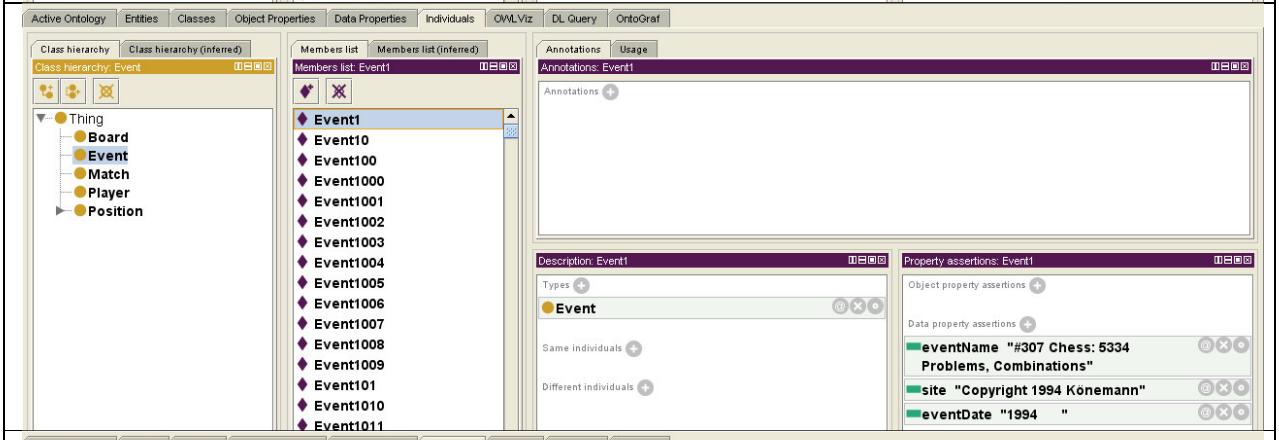
Tabla LVII: Pruebas Protégé (VI).

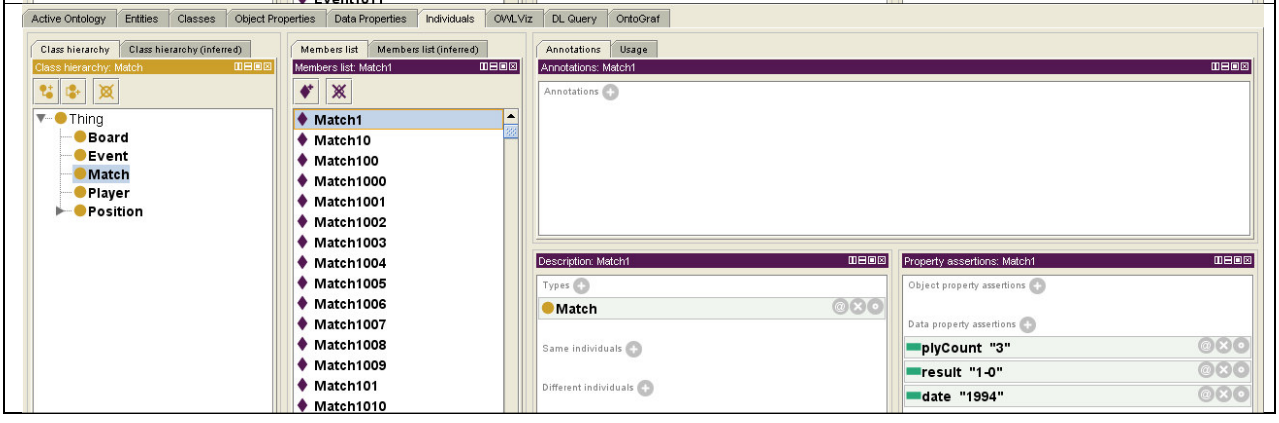
Fichero "MateInTwo.pgn":

[Event "#307 Chess: 5334 Problems, Combinations"]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in two"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "1Q6/8/8/8/8/k2K4/8/8 w - - 0 1"]
[PlyCount "3"]
[EventDate "1994.??.??"]

1. Kc3 {[%cal Rb8a7,Rb8b4,Rb4b2]} Ka2 (1... Ka4 2. Qb4#) 2. Qb2# 1-0







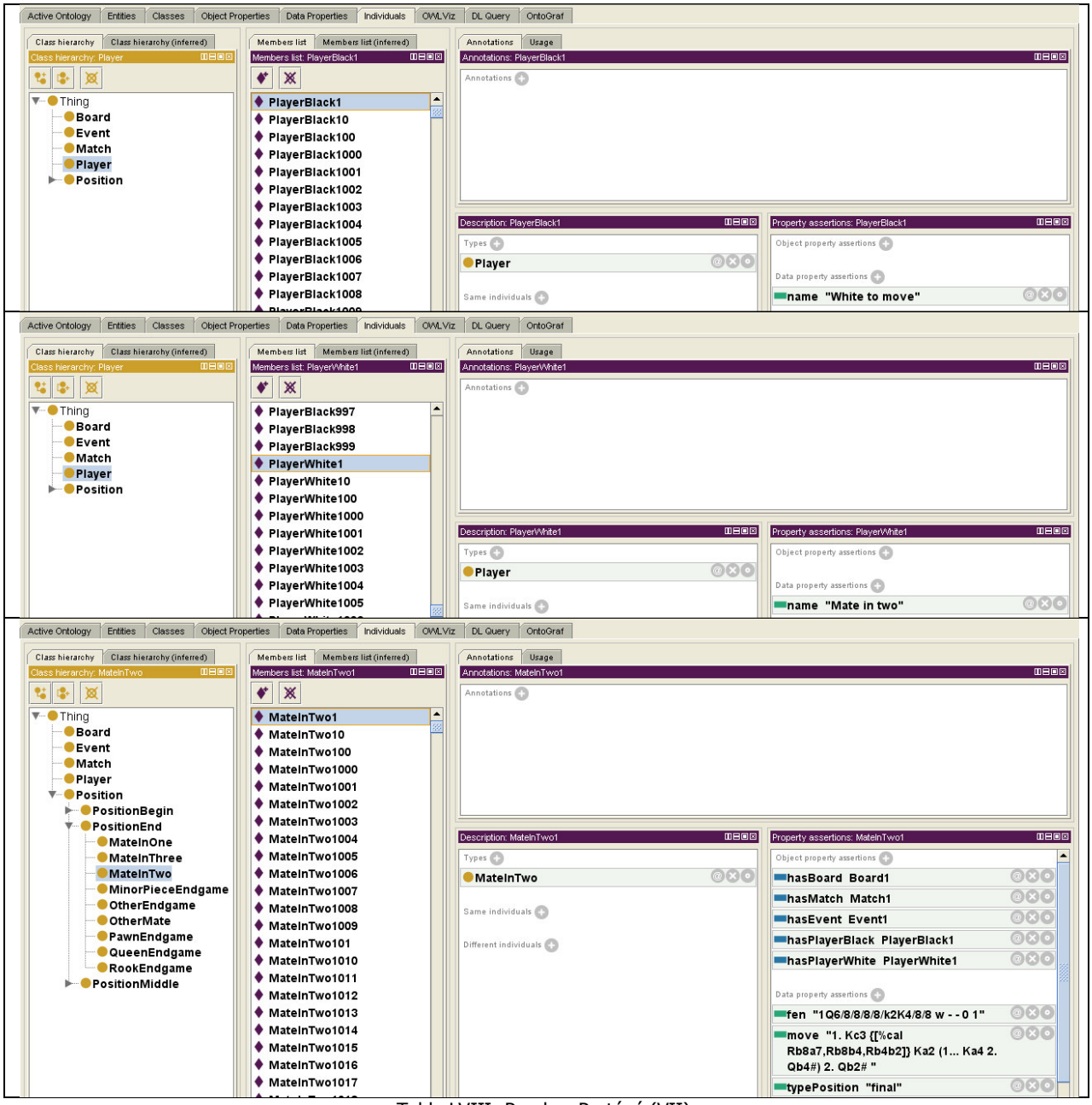
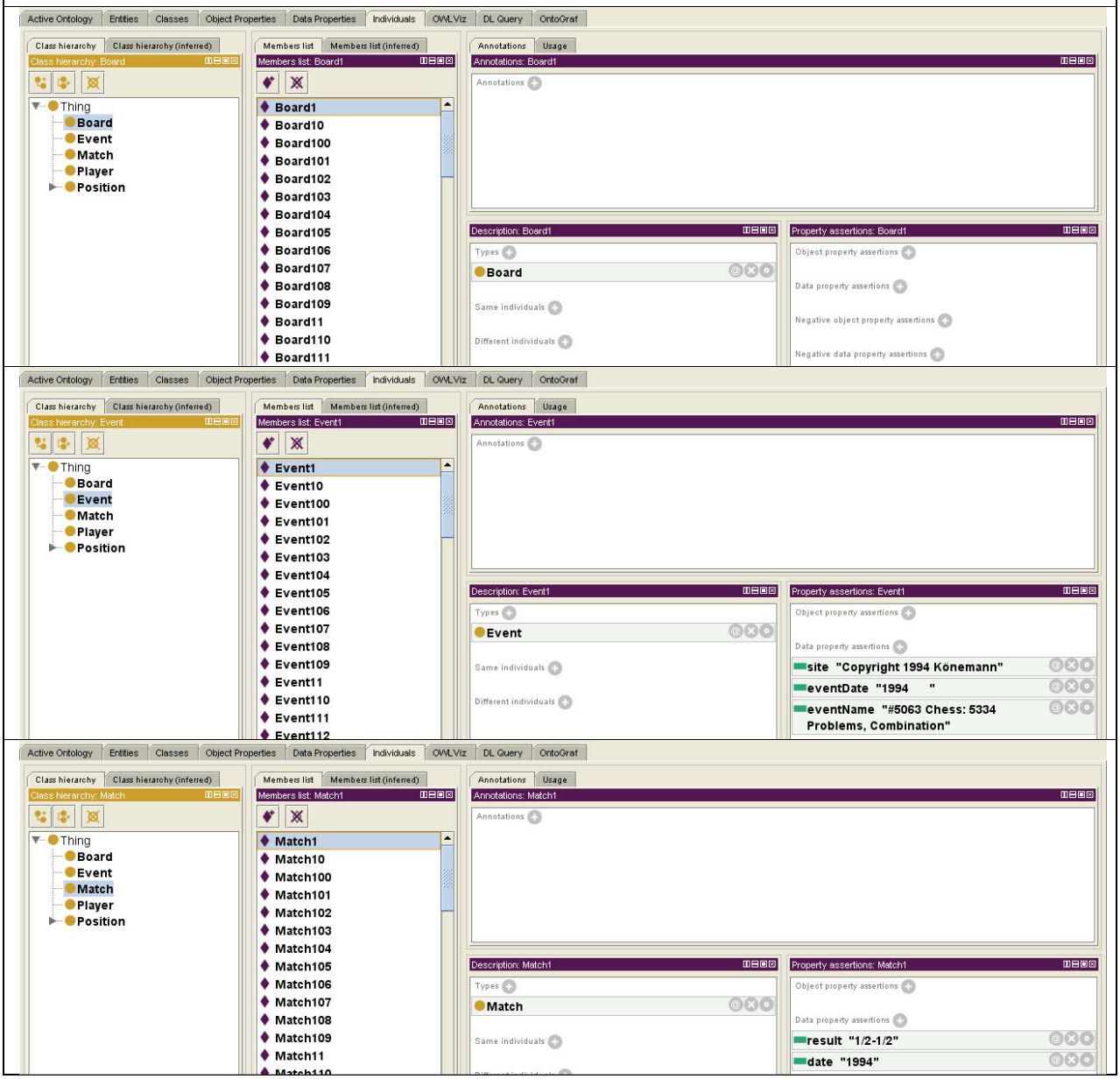


Tabla LVIII: Pruebas Protégé (VII).

Fichero "OtherEndgame.pgn":

```
[Event "#5063 Chess: 5334 Problems, Combination"]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Simple endgames"]
[Black "White to move"]
[Result "1/2-1/2"]
[SetUp "1"]
[FEN "4k3/4P3/3K4/8/8/8/8/8 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Ke6 {=} 1/2-1/2
```



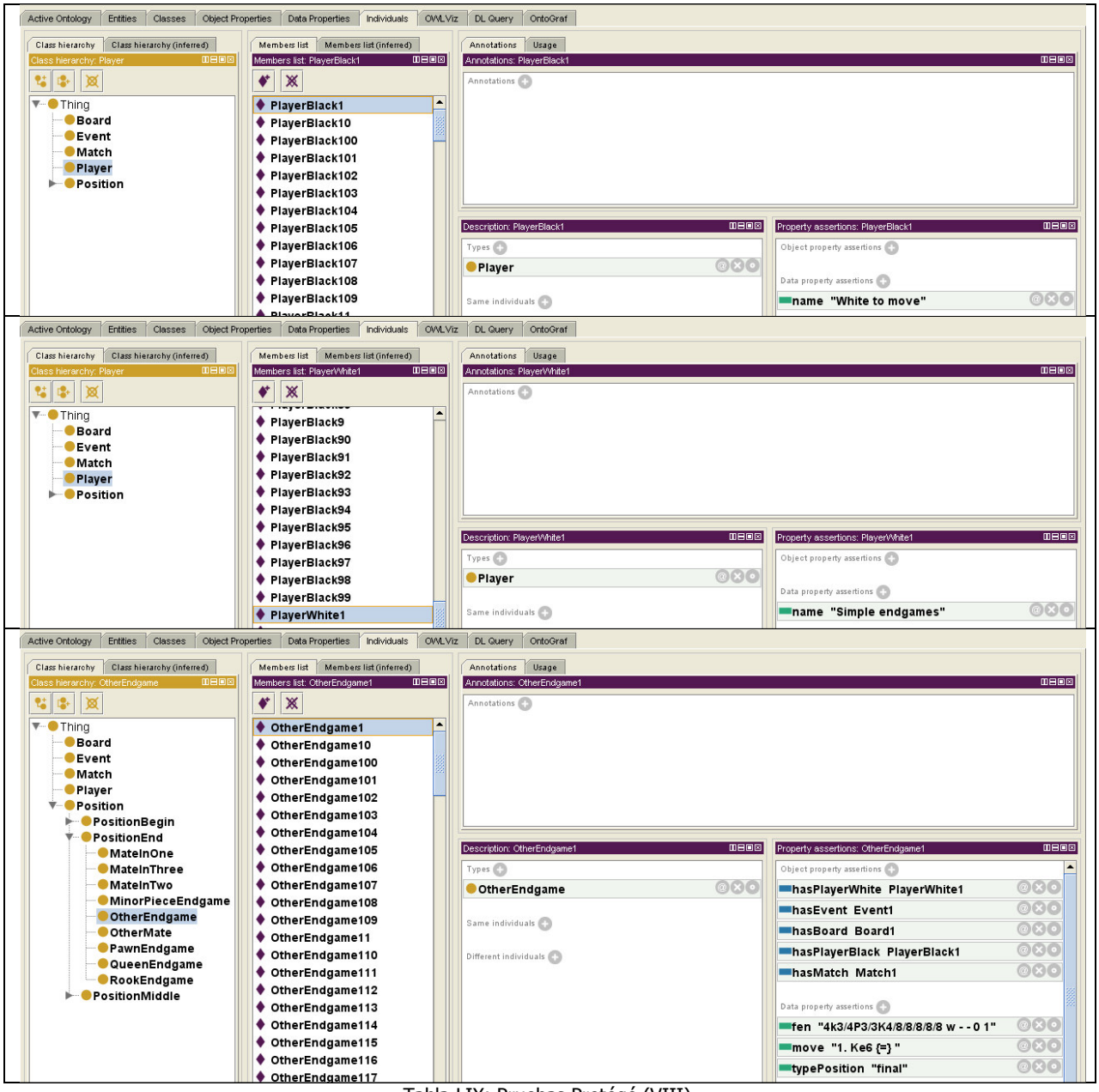
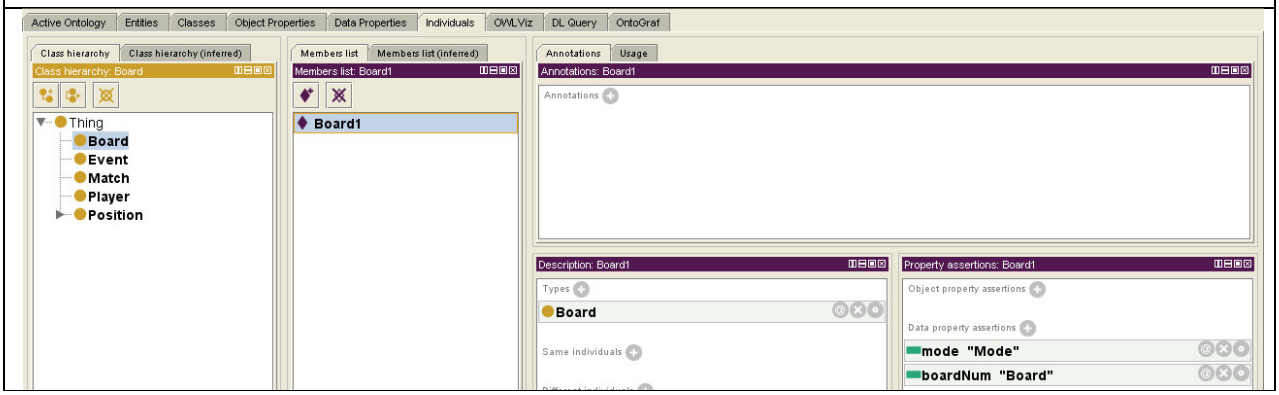
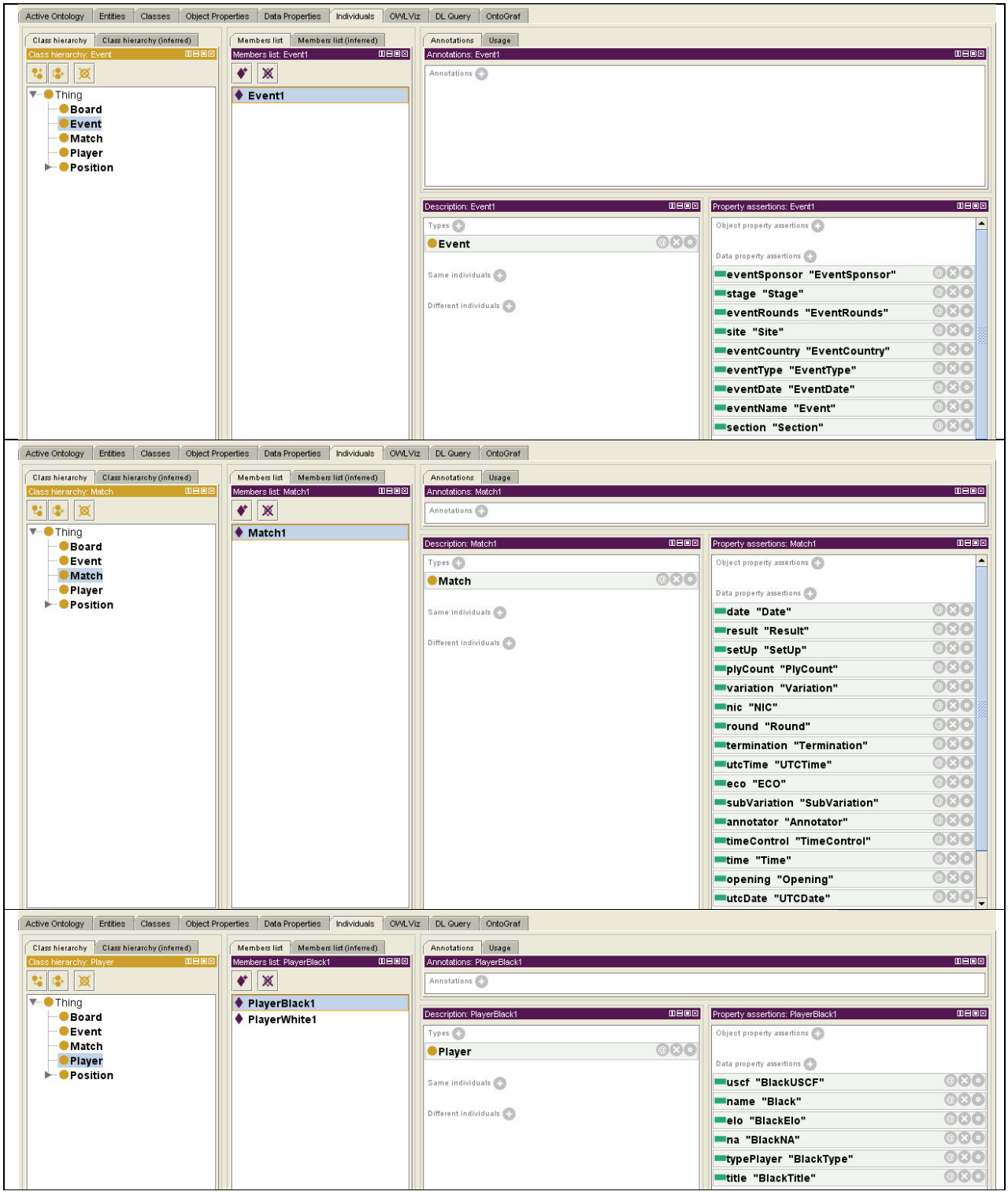


Tabla LIX: Pruebas Protégé (VIII).

Fichero "Prueba1.pgn":

[Event "Event"]
[Site "Site"]
[Date "Date"]
[Round "Round"]
[White "White"]
[Black "Black"]
[Result "Result"]
[WhiteTitle "WhiteTitle"]
[BlackTitle "BlackTitle"]
[WhiteElo "WhiteElo"]
[BlackElo "BlackElo"]
[WhiteUSCF "WhiteUSCF"]
[BlackUSCF "BlackUSCF"]
[WhiteNA "WhiteNA"]
[BlackNA "BlackNA"]
[WhiteType "WhiteType"]
[BlackType "BlackType"]
[EventDate "EventDate"]
[EventType "EventType"]
[EventCountry "EventCountry"]
[EventRounds "EventRounds"]
[EventSponsor "EventSponsor"]
[Section "Section"]
[Stage "Stage"]
[Board "Board"]
[Opening "Opening"]
[Variation "Variation"]
[SubVariation "SubVariation"]
[ECO "ECO"]
[NIC "NIC"]
[Time "Time"]
[UTCTime "UTCTime"]
[UTCDate "UTCDate"]
[TimeControl "TimeControl"]
[SetUp "SetUp"]
[FEN "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"]
[Termination "Termination"]
[Annotator "Annotator"]
[Mode "Mode"]
[PlyCount "PlyCount"]
1.move *




Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Event
Thing
Board
Event
Match
Player
Position
Members list
Members list (inferred)
Members list: Event1
Event1
Annotations
Usage
Annotations: Event1
Description: Event1
Types
Event
Same individuals
Different individuals
Property assertions: Event1
Object property assertions
Data property assertions
eventSponsor "EventSponsor"
stage "Stage"
eventRounds "EventRounds"
site "Site"
eventCountry "EventCountry"
eventType "EventType"
eventDate "EventDate"
eventName "Event"
section "Section"
Class hierarchy: Match
Members list: Match1
Match1
Annotations: Match1
Description: Match1
Match
Property assertions: Match1
date "Date"
result "Result"
setUp "SetUp"
plyCount "PlyCount"
variation "Variation"
nic "NIC"
round "Round"
termination "Termination"
utcTime "UTCTime"
eco "ECO"
subVariation "SubVariation"
annotator "Annotator"
timeControl "TimeControl"
time "Time"
opening "Opening"
utcDate "UTCDate"
Class hierarchy: Player
Members list: PlayerBlack1
PlayerBlack1
PlayerWhite1
Annotations: PlayerBlack1
Description: PlayerBlack1
Player
Property assertions: PlayerBlack1
uscf "BlackUSCF"
name "Black"
elo "BlackElo"
na "BlackNA"
typePlayer "BlackType"
title "BlackTitle"

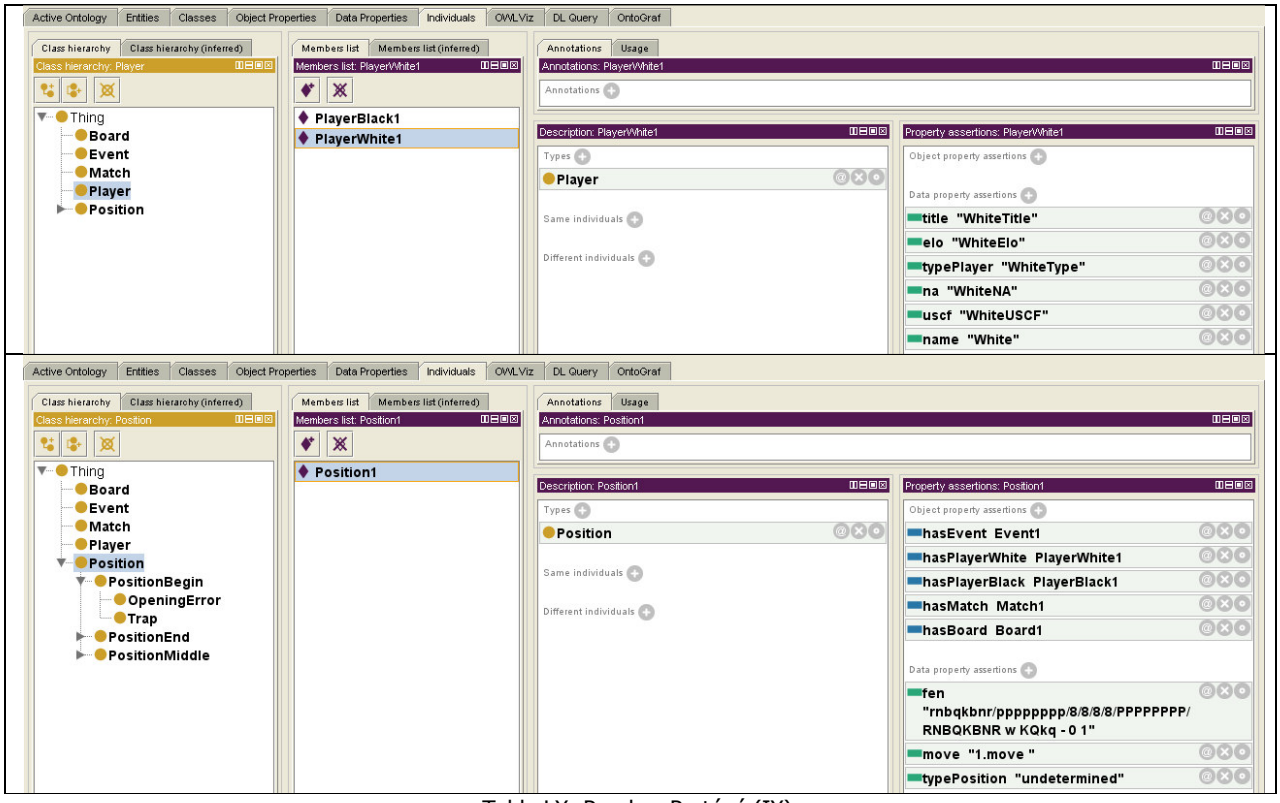
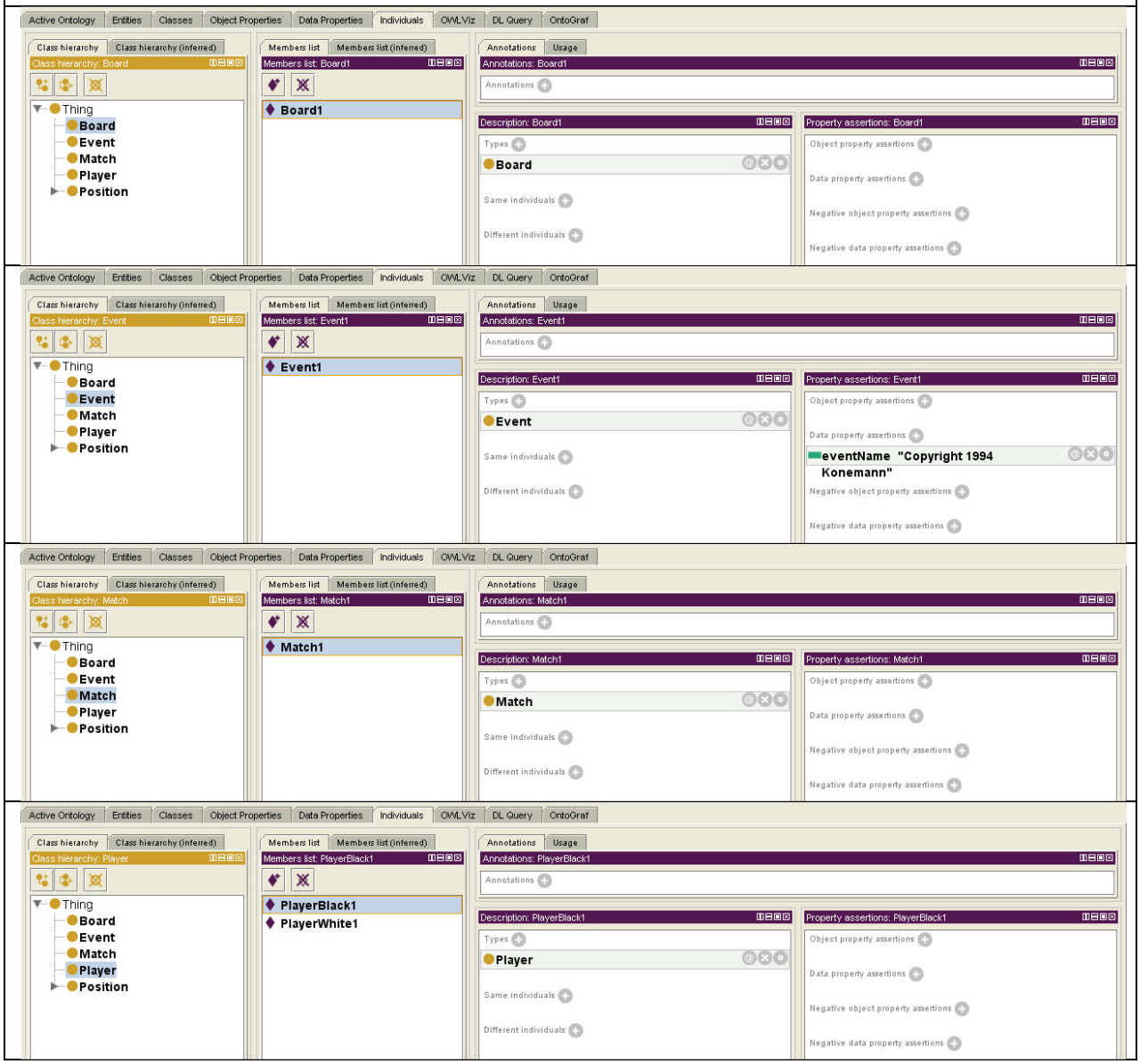


Tabla LX: Pruebas Protégé (IX).

Fichero "Prueba2.pgn":

[Event "Copyright 1994 Könemann"]
[FEN "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"]

1.e4 *



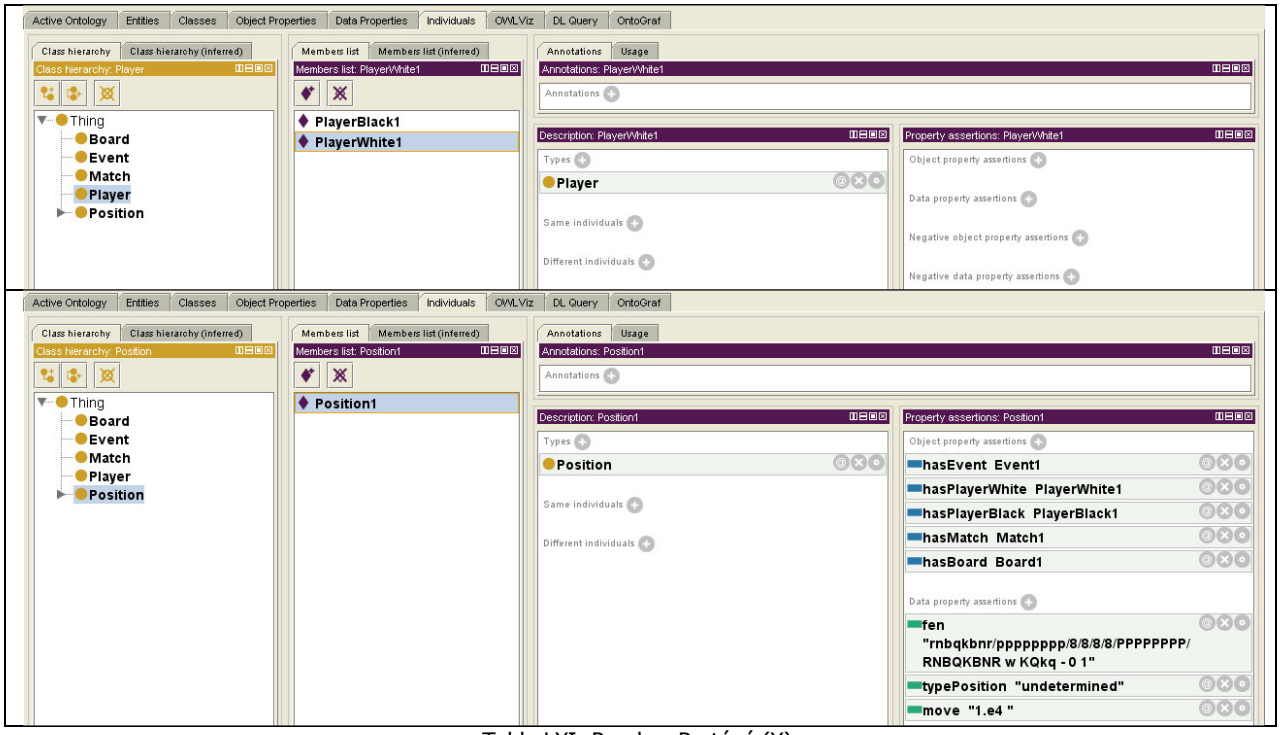
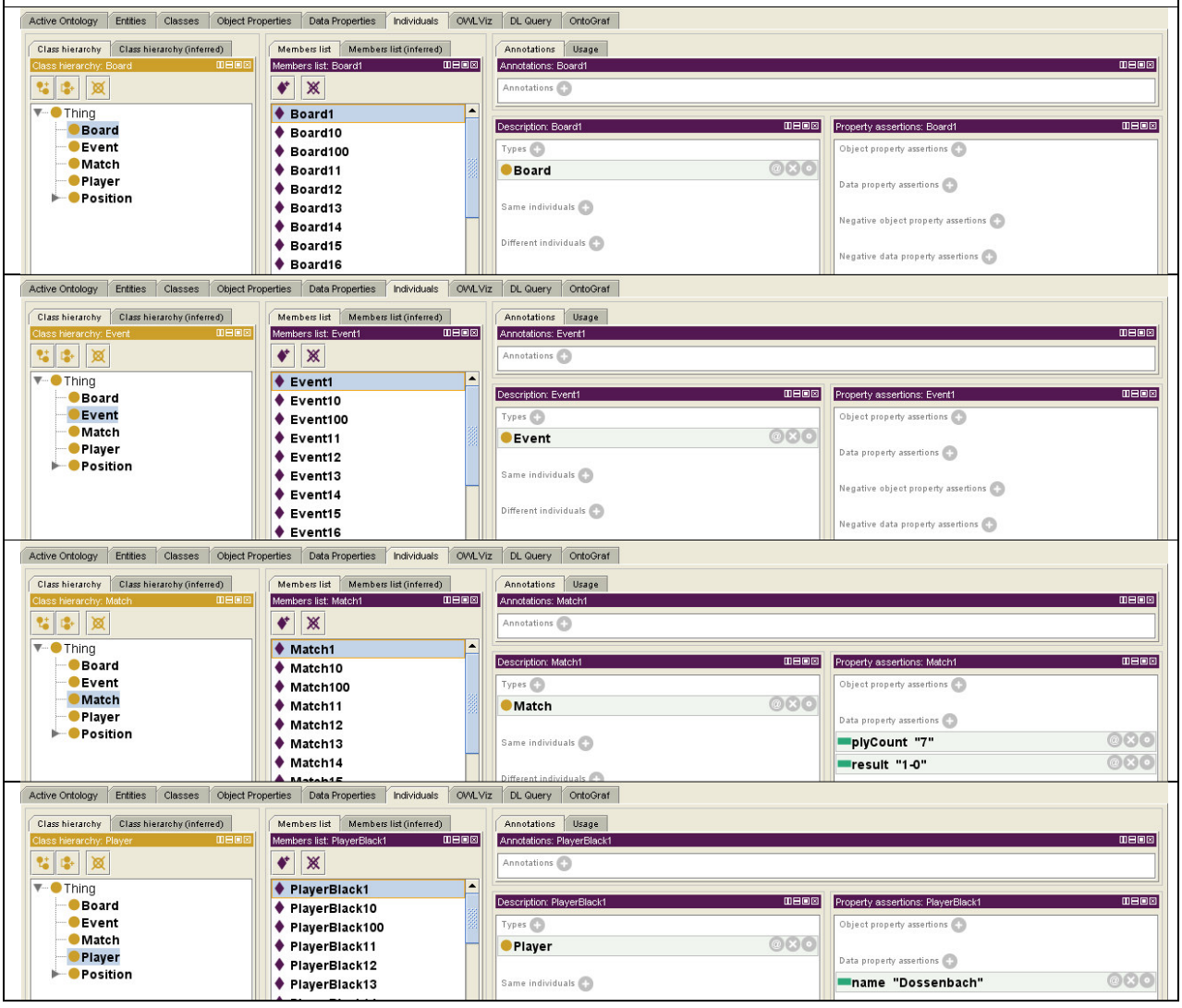


Tabla LXI: Pruebas Protégé (X).

Fichero "Tactica1.pgn":

[Event "?"]
[Site "?"]
[Date "????.??.??"]
[Round "?"]
[White "Hedinger"]
[Black "Dossenbach"]
[Result "1-0"]
[SetUp "1"]
[FEN "6k1/2b2p1p/ppP3p1/4p3/PP1B4/5PP1/7P/7K w - - 0 1"]
[PlyCount "7"]

1. Bxb6 Bxb6 2. a5 Bc7 (2... Bd8 3. b5 axb5 (3... Bxa5 4. bxa6 $16) 4. a6 $16)
3. b5 axb5 4. a6 1-0



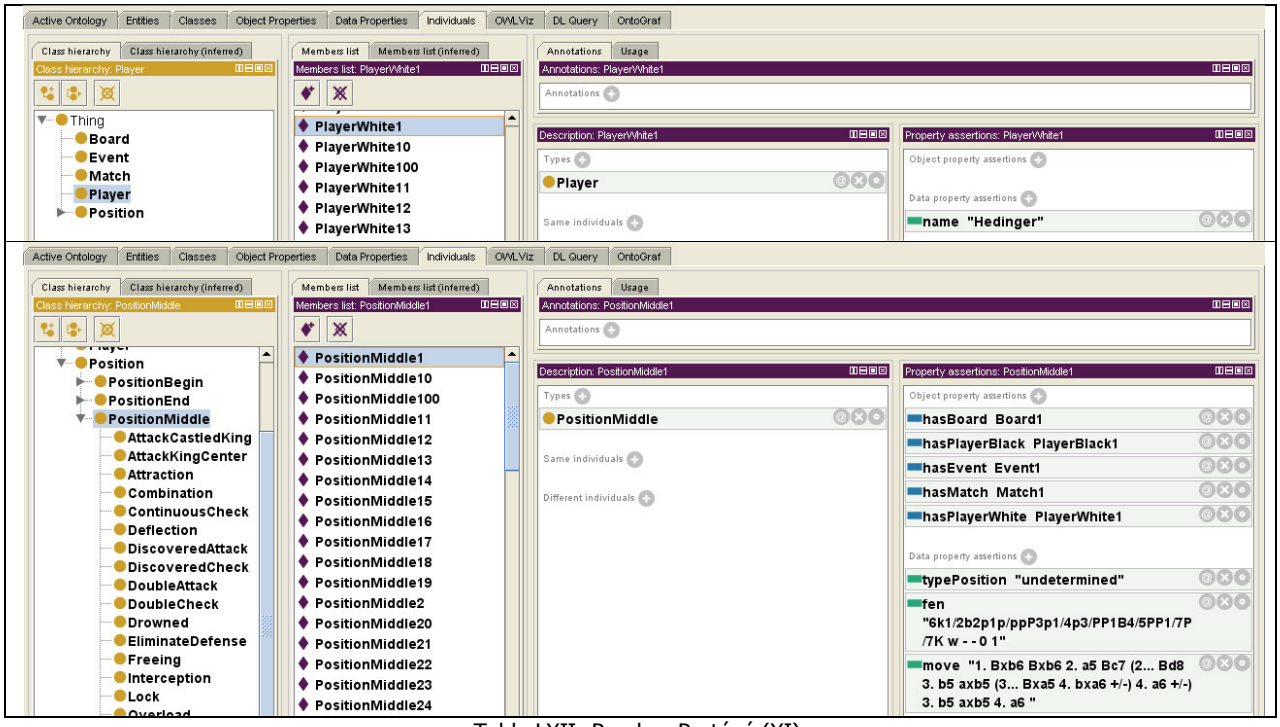
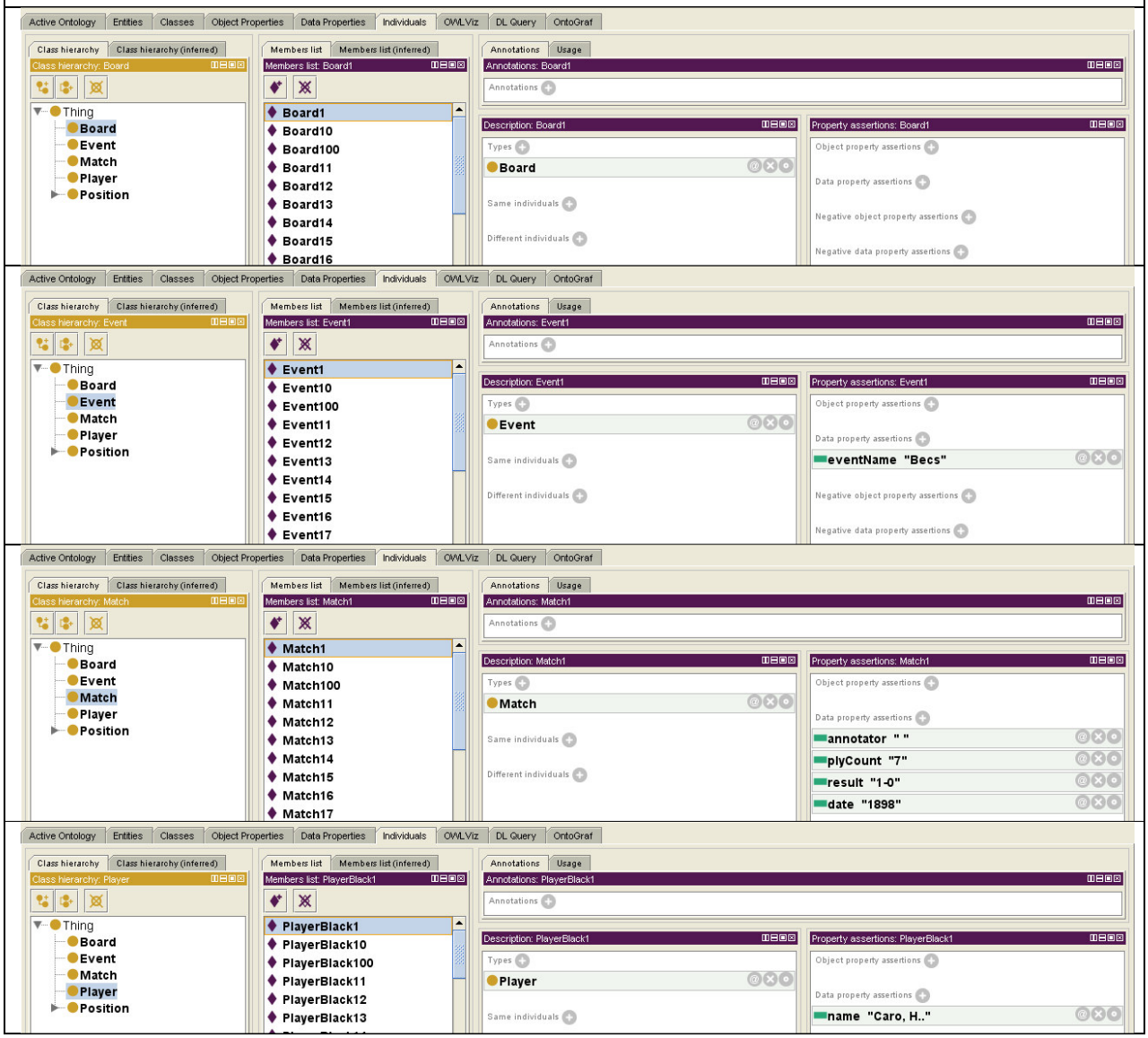


Tabla LXII: Pruebas Protégé (XI).

Fichero "Tactica2.pgn":

[Event "Becs"]
[Site "?"]
[Date "1898.??.??"]
[Round "?"]
[White "Tschigorin, Mikhail"]
[Black "Caro, H.."]
[Result "1-0"]
[Annotator " "]
[SetUp "1"]
[FEN "2r1k3/6pr/p1nBP3/1p3p1p/2q5/2P5/P1R4P/K2Q2R1 w - - 0 1"]
[PlyCount "7"]

1. Rxg7 Rxg7 (1... Rh8 2. Qxh5+ Rxh5 (2... Kd8 3. Qxh8#) 3. Rg8#) (1... Qxe6 2. Rxh7 $16) 2. Qxh5+ Kd8 (2... Rf7 3. Qxf7+ Kd8 4. Qf8#) 3. Qh8+ Rg8 4. Qxg8# 1-0



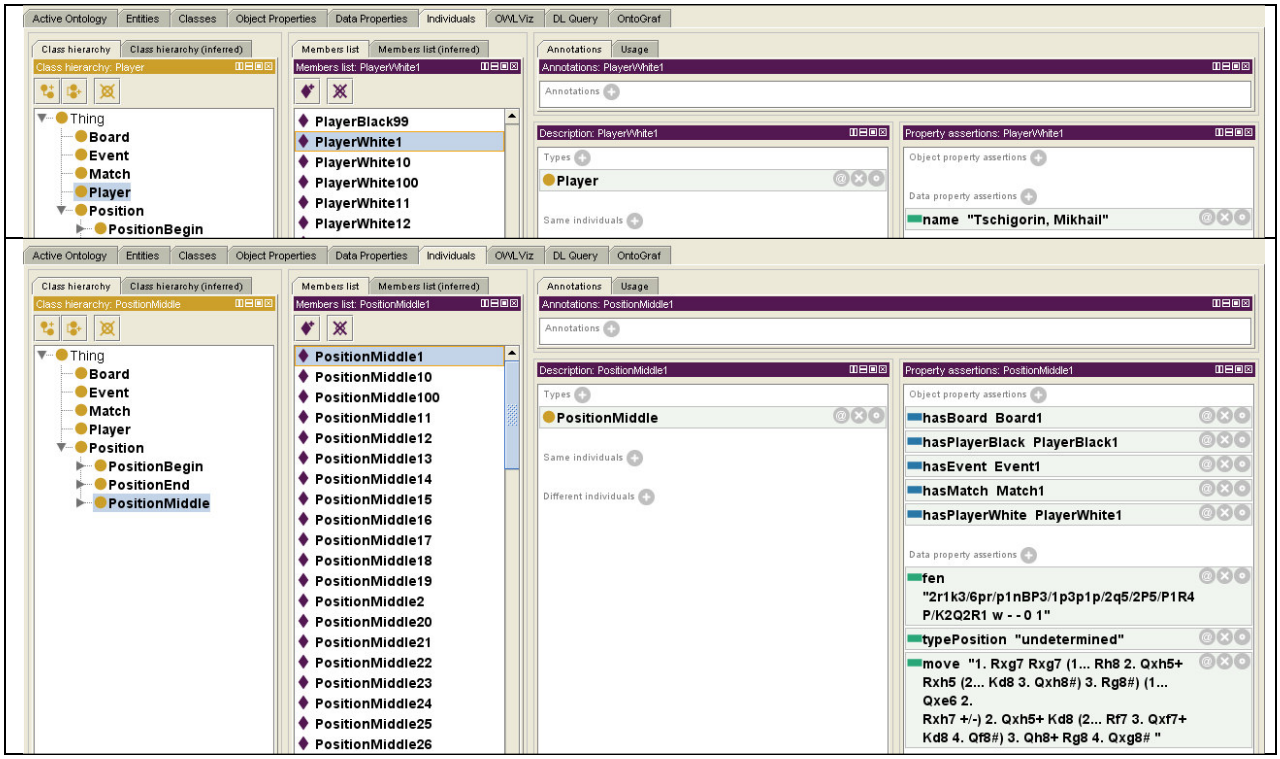
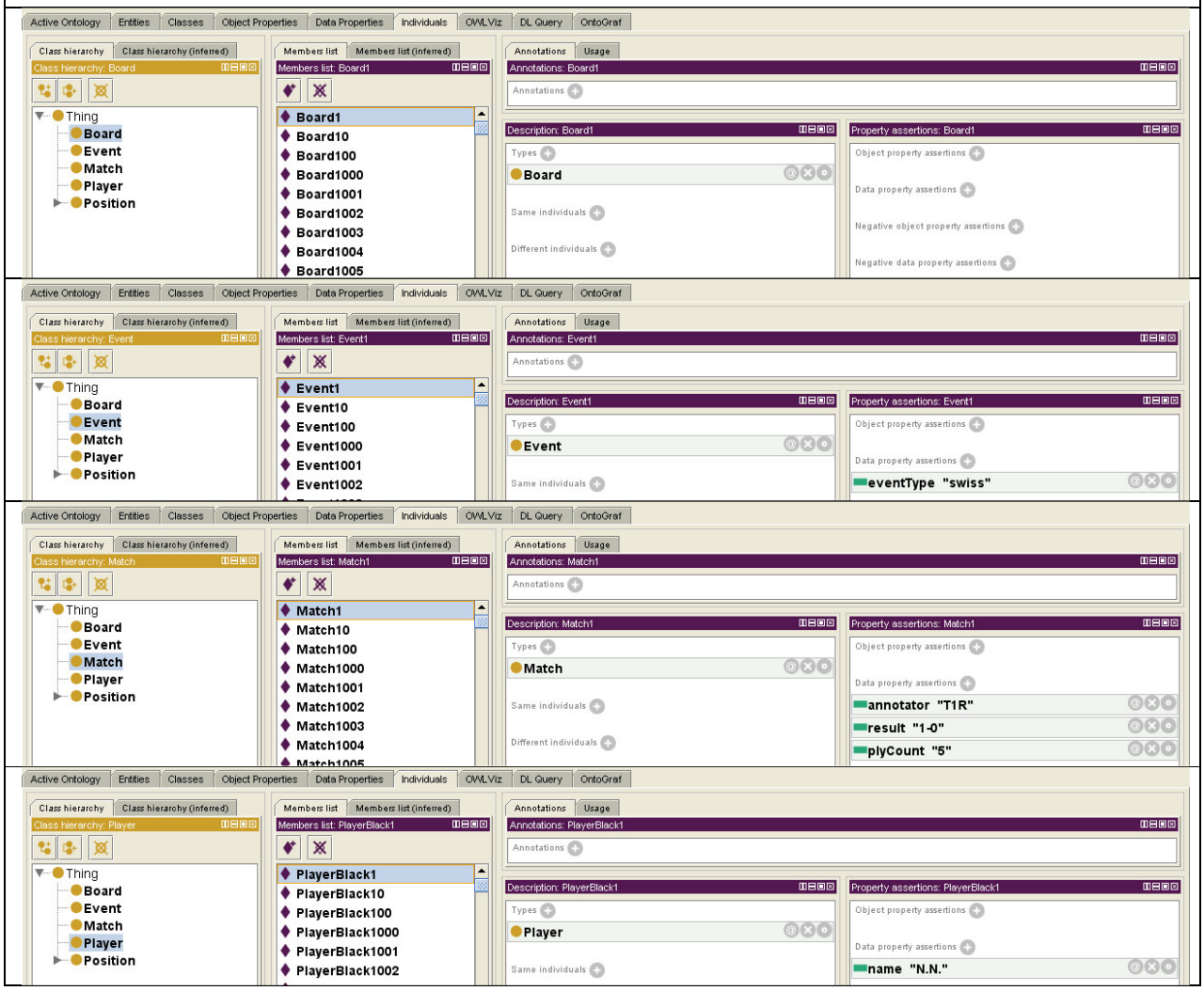


Tabla LXIII: Pruebas Protégé (XII).

Fichero "TacticalAuerswald1.pgn":

[Event "?"]
[Site "?"]
[Date "????.??.??"]
[Round "?"]
[White "N.N."]
[Black "N.N."]
[Result "1-0"]
[Annotator "T1R"]
[SetUp "1"]
[FEN "2r2rk1/pp1bqpp1/2nppn1p/2p3N1/1bP5/1PN3P1/PBQPPPBP/3R1RK1 w - - 0 1"]
[PlyCount "5"]
[EventType "swiss"]

1. Nd5 exd5 (1... hxg5 2. Nxe7+ Nxe7) 2. Bxf6 hxg5 3. Bxe7 1-0



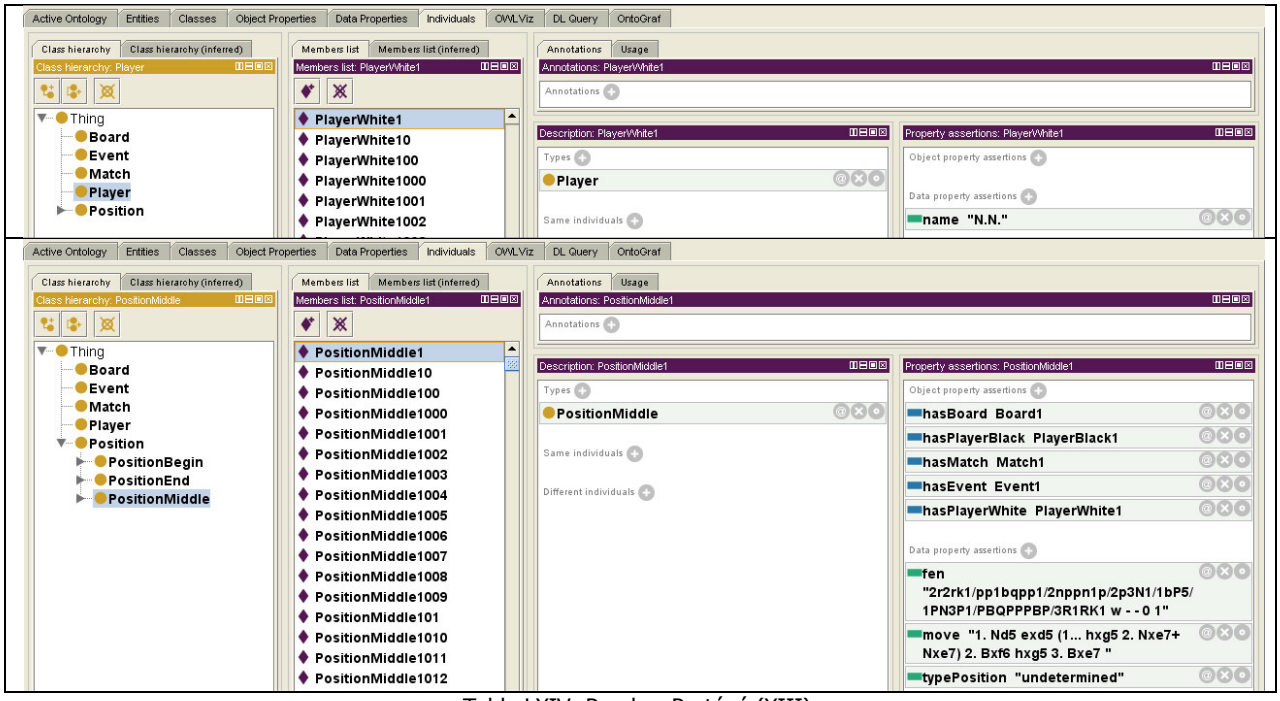
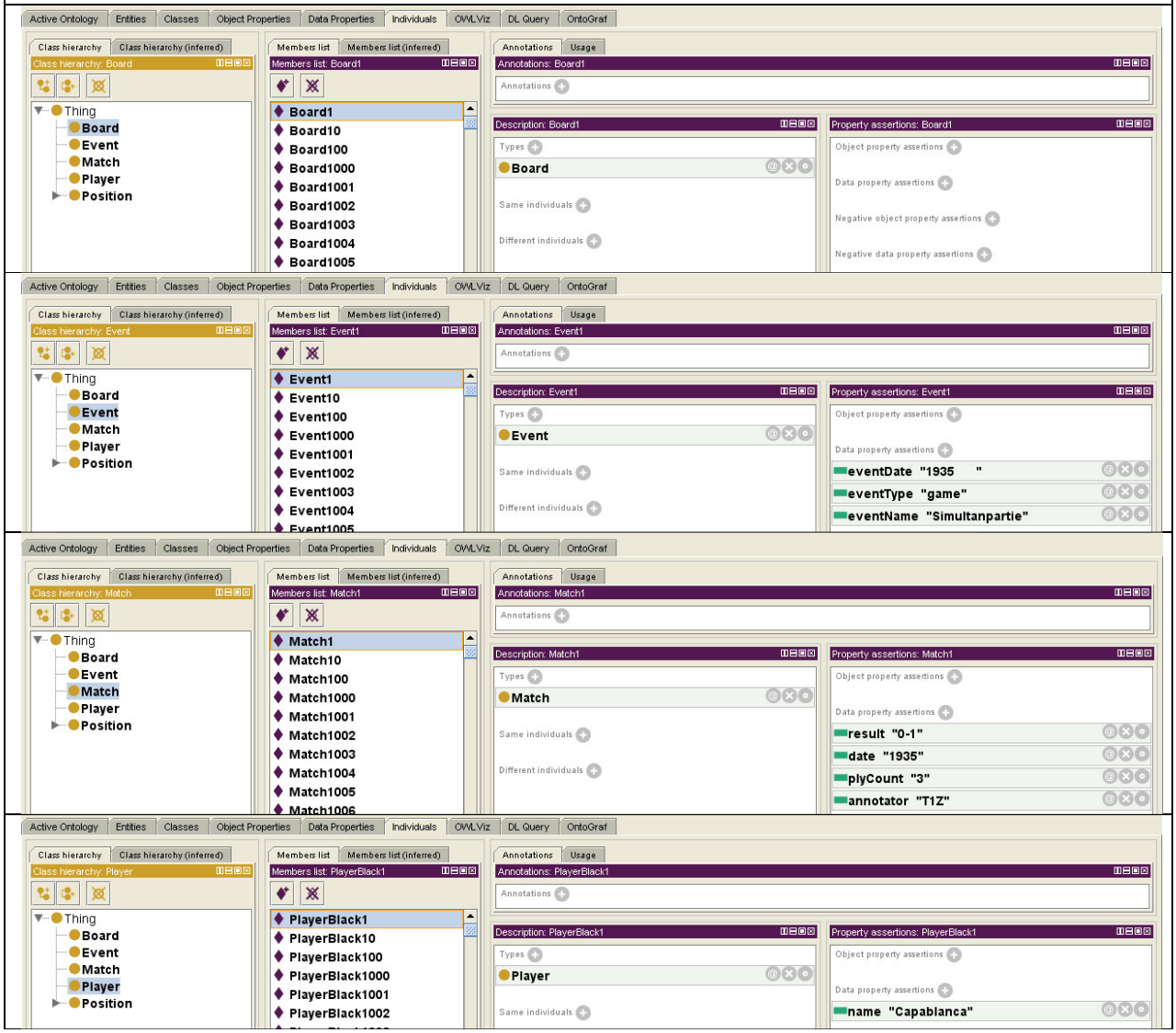


Tabla LXIV: Pruebas Protégé (XIII).

Fichero "TacticalAuerswald2.pgn":

[Event "Simultanpartie"]
[Site "?"]
[Date "1935.??.??"]
[Round "?"]
[White "N.N."]
[Black "Capablanca"]
[Result "0-1"]
[Annotator "T1Z"]
[SetUp "1"]
[FEN "r3kbnr/ppp2ppp/8/q2Ppb2/1n6/2N5/PPPBNPPP/2RQKB1R b - - 0 1"]
[PlyCount "3"]
[EventDate "1935.??.??"]
[EventType "game"]

1... Bxc2 2. Rxc2 Nd3# 0-1



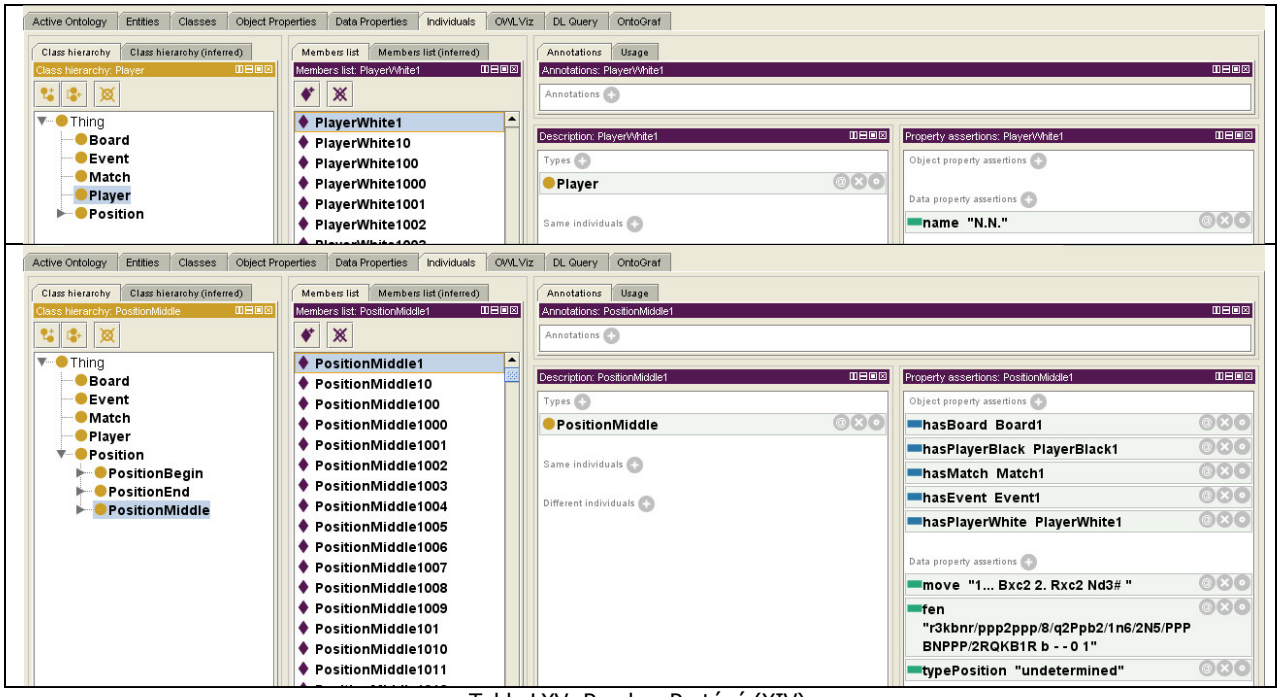
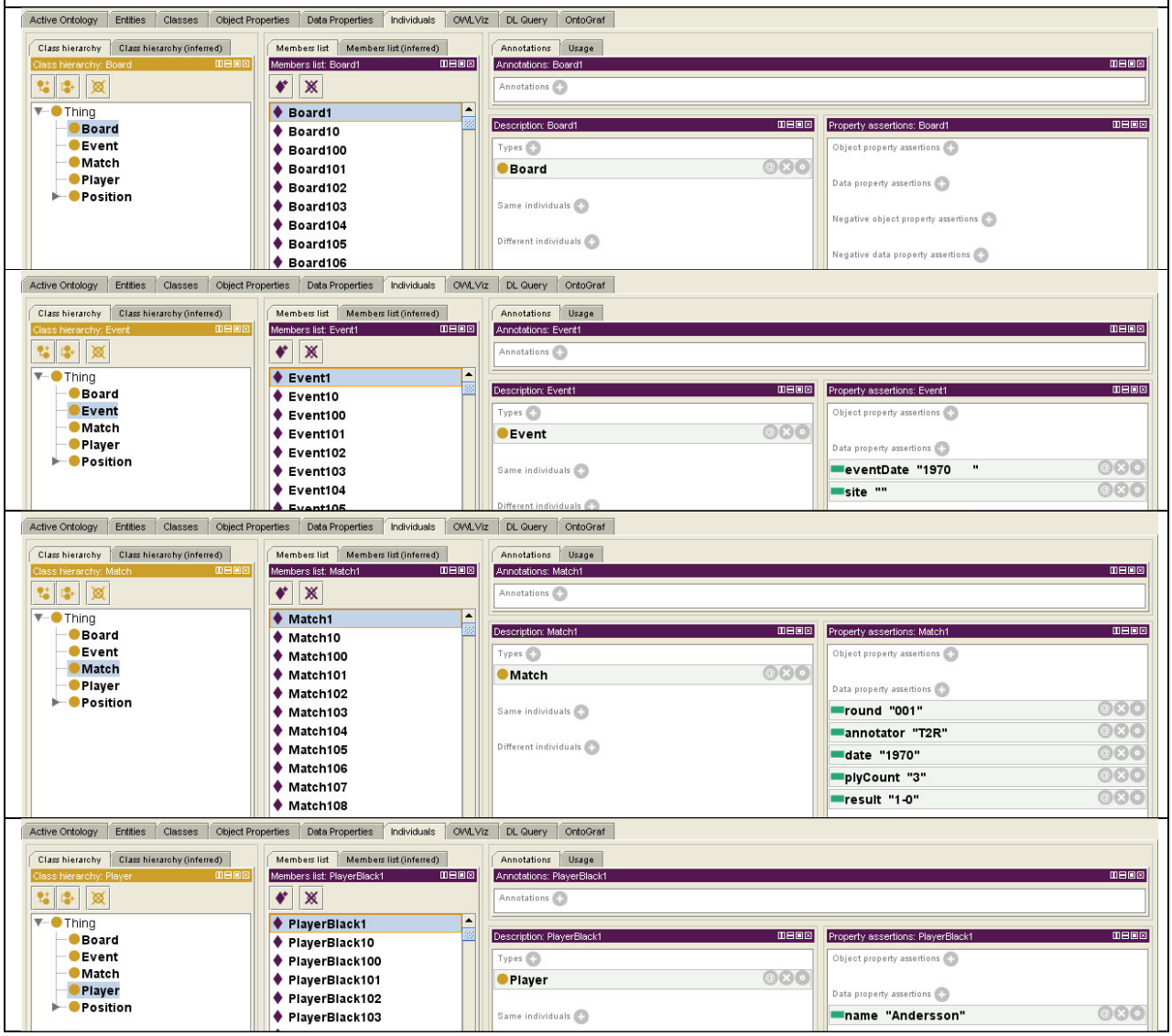


Tabla LXV: Pruebas Protégé (XIV).

Fichero "tactics.pgn":

[Event "?"]
[Site ""]
[Date "1970.??.??"]
[Round "001"]
[White "Fischer"]
[Black "Andersson"]
[Result "1-0"]
[Annotator "T2R"]
[EventDate "1970.??.??"]
[SetUp "1"]
[FEN "5q1k/2p3Rp/1p1rnP2/p4p2/2Q5/PP2P3/1B3P1P/7K w - - 0 1"]
[PlyCount "3"]

1. Qxe6 Rxe6 2. f7 h6 3. Rg8+ Kh7 4. Rxf8 Kg6 1-0



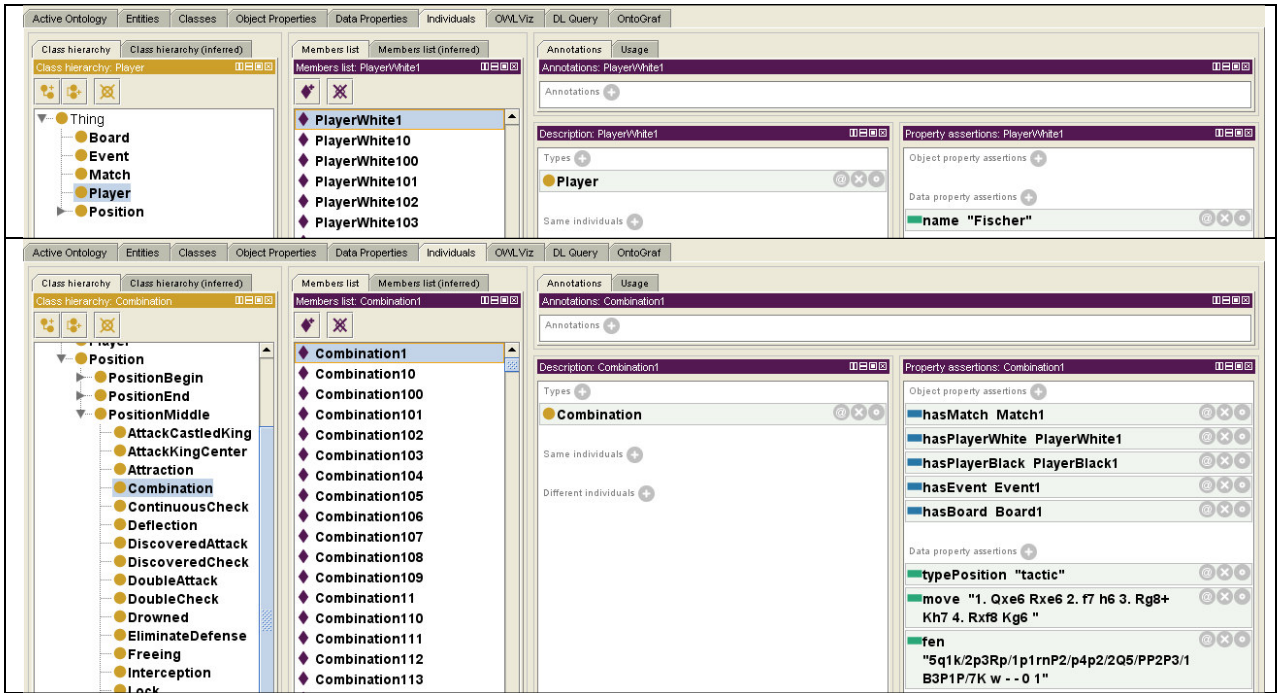


Tabla LXVI: Pruebas Protégé (XV).

Como hemos podido ver, la primera posición de cada fichero PGN está bien construida en el sentido de que toda la información contenida en las etiquetas y en la solución de la posición está también contenida en las propiedades de cada individuo.

3.2 Siguientes pasos

Cerramos este capítulo con algunos comentarios sobre posibles puntos de mejora del proceso presentado, no realizados por considerar que no forman parte de los objetivos básicos presentados para este PFC.

Resultados en otros idiomas

Dado que los ficheros fuente tratados tenían nomenclatura FEN en inglés, y dado que la ontología definida y salida elaborada también han respetado este idioma, se han mantenido dicho idioma en la representación de los movimientos solución de cada una de las posiciones propuestas. Incluso las salidas en la ejecución del generador, aún siendo en castellano, se han realizado sin "eñes" y sin acentos. En una nueva versión del programa se podría incluir como parámetro de entrada el idioma que se desea utilizar como salida de información. Para poder incluir esta funcionalidad se requiere ampliar el analizador léxico y tratar de forma adecuada cada uno de los lexemas contenidos en la sección de movimientos. El proceso se complica un poco, pero los beneficios son considerables, dado que también se podría generar "dinamismo" (movimiento de piezas) con los movimientos indicados en la solución.

Además, habría que revisar el método de declaración de la ontología con Protégé dado que según se ha declarado, cualquier ontología que incluya acentos, "eñes", o cualquier otro carácter no incluido en el juego de caracteres básico.

Código HTML y OWL integrado

La versión del proceso descrita y desarrollada en este PFC contempla la generación de dos ficheros independientes, el primero contiene el código HTML que presenta las posiciones de ajedrez, sus soluciones y los datos complementarios. El segundo fichero conforma la base de conocimiento.

Una nueva versión del programa complementaria, podría generar un único fichero HTML con contenido OWL en su interior.

Se ha optado por la primera versión por ser considerada mucho más didáctica y sencilla de validar, pero hubiese sido igualmente válida la segunda versión, y no hubiese requerido un esfuerzo mayor.

Base de conocimiento única

En cada una de las ejecuciones que hemos presentado en el apartado anterior se ha creado una base de conocimiento con una parte común, las clases y propiedades, y unas instancias dependientes del contenido de cada fichero PGN. El siguiente paso a realizar sería la unificación de toda esta información en un único fichero, el cual sería pasado a un entorno web para que pudiese ser utilizado como base de conocimiento de referencia de nuestra ontología.

Según está el proceso desarrollado no se podría realizar dicha unificación, dado que nos encontraríamos instancias con el mismo nombre. Por ejemplo, el IRI "Board1" aparecerá en todos los ficheros OWL generados. Para evitar este problema se podría incluir un prefijo en cada IRI que podría

coincidir con el nombre del fichero tratado, o podría ser igualmente un valor recuperado como argumento de entrada.

Ampliación HTML5

En este PFC hemos dejado patente las grandes ventajas de algunas características del nuevo lenguaje HTML5. En concreto, se ha visto que el elemento "canvas" ofrece una mejora significativa a la hora de presentar imágenes dinámicas en pantalla. Pero no es la única nueva característica que podría ser explotada. Por ejemplo, se podría incluir características de almacenamiento local de información, para que el usuario de la página estuviese informado del número de veces que ha intentado resolver un ejercicio, las veces que lo ha resuelto correctamente, o no. Se podrían incluir algunos botones adicionales que permitiesen indicar al usuario dicha información, que sería almacenada localmente. También se podrían incluir nuevos elementos gráficos como, por ejemplo, un reloj que indicara el tiempo total previsto para la resolución del ejercicio, y el tiempo transcurrido.

Ampliación base de conocimiento

Este PFC ha sido probado con un buen número de posiciones didácticas, más de 15.000 ejercicios se presentan junto a esta memoria, pero sólo para ciertos casos estratégico-tácticos. Para desarrollar una base de conocimiento completa sería necesario utilizar un fichero PGN para cada una de las clases definidas en la ontología. El único problema para realizar esta tarea es la búsqueda de bases de datos válidas a cada posición (con derechos de autor en la mayoría de las ocasiones) dado que para la generación de las nuevas bases de conocimiento y páginas HTML simplemente se requiere la nueva ejecución del generador GPAHS.

Análisis semántico

La versión del generador presentada ofrece un analizador semántico muy elemental, realizando básicamente la eliminación de valores cuando se utiliza el carácter comodín "?" en el fichero PGN, y simplificando los campos fecha cuando en los mismos se incluyen también los caracteres comodín "??".

Una de las posibles mejoras a realizar sería la ampliación de las validaciones sobre cada uno de los campos obtenidos del fichero PGN. Por ejemplo, cuando se indica un país donde se realiza un evento ajedrecístico se podría validar que el código introducido se corresponde con alguno de los códigos normalizados por el Comité Olímpico Internacional (COI).

4 REFERENCIAS

Las referencias en las que nos hemos apoyado para el desarrollo de esta memoria han sido las siguientes:

Análisis y diseño de generación de código:

- JLex / CUP:
  - Material proporcionado en la asignatura de Compiladores II. En concreto, material "Jex y CUP Herramientas para la generación automática de analizadores léxicos y sintácticos" de Francesc Bagés Vaqué.

Formato PGN:

- Descripción de la notación Portable Game Notation "PGN":
  - http://es.wikipedia.org/wiki/Notaci%C3%B3n_portable_de_juego
  - http://www.chessclub.com/help/PGN-spec
- Ficheros con posiciones PGN utilizadas en las pruebas integradas:
  - http://elblogdecaissa.blogspot.com/

Formato FEN:

- Descripción del formato FEN:
  - http://ajedrezmagico.blogspot.com/2008/02/notacion-fen.html
  - http://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation

Lenguaje OWL y web semántica:

- Visión general del lenguaje de ontologías web (OWL), de la W3C:
  - http://www.w3.org/2007/09/OWL-Overview-es.html
- Guía OWL, de la W3C:
  - http://www.w3.org/TR/owl-guide/
- Definición formal OWL, de la W3C:
  - http://www.w3.org/TR/owl-semantics/
- Tesis doctoral "Ontology Transformations Between Formalisms" de Petr Aubrecht
  - http://www.google.es/url?sa=t&source=web&cd=7&ved=0CD0QFjAG&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.85.4488%26rep%3Drep1%26type%3Dpdf&ei=ydKITfO_G5H64AaIoY3wDQ&usg=AFQjCNFKDC3RsDPFlUyHK2V5obkxC_SrXw&sig2=ooNnfmap1fLEQJZ75hl4nA
- Proyecto Final de Carrera de Juan Miguel Larrayoz y Juan Ángel Larrayoz, del Instituto Universitario Autónomo del Sur, año 2009
  - http://www.google.es/url?sa=t&source=web&cd=7&ved=0CC8QFjAG&url=http%3A%2F%2Fjftstool.googlecode.com%2Fsvn-history%2Fr482%2Ftrunks%2FAnalisis%2Fcarpeta%2FAnexos.doc&ei=mDLRTdKeD8jD8QOh8eHfDQ&usg=AFQjCNFXhSNGg8e5yvp5FxmJgcWi0L8I-g

- Estudio de OWL y su integración con Protégé, de Fernando Leandro Baladrón y Salvador Rubio Montero:
  - http://personales.alumno.upv.es/ferleaba/documents/OWL_Protege.pdf
- RDF Schema:
  - http://es.wikipedia.org/wiki/RDF_Schema
- Libro "*Manual de Web Semántica*" de Grigoris Antoniou y Fran van Harmelen, ISBN: 978-84-9836-780-5.
- Artículo de Tim Berners-lee, sobre Web Semántica:
  - http://www.dcc.uchile.cl/~cgutierr/cursos/IC/semantic-web.pdf

RFD Schema:

- http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

Editor Protégé:

- Primeros pasos con Protégé:
  - http://protege.stanford.edu/publications/GettingStarted.pdf
  - http://protege.stanford.edu/overview/protege-owl.html
- Guía de usuario Protégé:
  - http://protege.stanford.edu/publications/UserGuide.pdf
- Documentación oficial en castellano sobre la aplicación Protégé
  - http://protege.stanford.edu/publications/ontology_development/ontology101-es.pdf

Lenguaje HTML5:

- Wiki HTML5:
  - http://es.wikipedia.org/wiki/HTML_5
- Especificaciones:
  - http://dev.w3.org/html5/spec/Overview.html
- Web Schools:
  - http://www.w3schools.com/html5/default.asp
- HTML 5 Doctor:
  - http://html5doctor.com/understanding-aside/
- Libro "*HTML5 and CSS3 Develop with Tomorrow's Standards Today*" de Brian P. Hogan – ISBN-10: 1-934356-68-9
- Libro "*The Essential Guide to HTML5: Using Games to learn HTML5 and JavaScript*", de Jeanine Meyer. ISBN: ISBN-10: 1-4302-3383-4, ISBN-13: 978-1-4302-3383-1

Herramientas de tratamiento de ficheros PGN:

- Utilidad Palmview:
  - http://www.enpassant.dk/chess/palview/index.htm
  - http://www.enpassant.dk/chess/palview/p3demo/normal1.htm
- Utilidad pgn2html:
  - http://www.muewer.homepage.t-online.de/pgn2html.html (nota: durante el desarrollo de este PFC la web no estaba disponible, pero hemos podido obtener información utilizando otros enlaces).
- Utilidad "PGN to JS":

  - http://www.mailchess.de/pgntojse.html (nota: durante el desarrollo de este PFC la web oficial no estaba disponible, pero hemos podido obtener información utilizando el siguiente enlace).
  - http://downloads.ugo.com/downloads/sharewares/tools/PGNtoJS_2.51.shtml
- Utilidad "LT-Pgn-Viewer":
  - http://www.lutanho.net/pgn/pgnviewer.html
- Utilidad PGN Reader Java Applet:
  - http://www.stonkie.com/en/pgnreader/index.html

Otras referencias:

- Leyes de ajedrez actuales de la FIDE:
  - http://www.feda.org/leyes/Leyes_2009.pdf
- Herramientas de libre distribución de tratamiento de ficheros PGN:
  - http://www.enpassant.dk/chess/diaeng.htm

5 ANEXOS

Dividimos los anexos que complementan el PFC en dos apartados:

Contenido	Apartados	Comentarios
Información complementaria	5.1	Anexos que complementan la información presentada en el PFC.
Implementación	5.2	Código fuente completo de todos los elementos desarrollados en este PFC.

Tabla LXVII: Apartados del capítulo 5.

5.1 Información complementaria

Los anexos contenidos en este apartado son los siguientes:

Contenido	Sección	Comentarios
Las leyes de ajedrez	5.1.1	En este anexo se presentan las leyes del ajedrez, información básica para comprender algunos puntos de este PFC.
Tácticas y estrategias ontológicas	5.1.2	En este anexo se describen las tácticas y estrategias ajedrecísticas que han sido incluidas en la definición de clases de nuestra ontología. Aparecen tanto los términos en inglés como en español.
Notación NAG	5.1.3	La notación NAG (Numeric Annotation Glyphs) permite simplificar los comentarios o anotaciones de las partidas, de una manera normalizada.
Siglas de países del Comité Olímpico Internacional (COI)	5.1.4	Algunos campos de los ficheros PGN contienen estas siglas de países, por ello, la inclusión de este último anexo.

Tabla LXVIII: Secciones del apartado 5.1.

NOTA: *Para reducir el número de páginas de la memoria entregada se ha reducido el tamaño de letra de todos estos anexos, así como su interlineado.*

5.1.1 Las Leyes del ajedrez

A continuación se incluyen, a modo de anexo, las leyes de ajedrez de la FIDE (Federación Internacional de Ajedrez) que se adoptaron en el 75º Congreso de la FIDE celebrado en Calvià (Mallorca) en octubre de 2004, entrando en vigor el 1 de Julio de 2005. Tras este reglamento no se han incluido nuevas normas relacionadas con el juego en sí.

Artículo 1: Naturaleza y objetivos de la partida de ajedrez

1.1.- La partida de ajedrez se juega entre dos adversarios que mueven alternativamente sus propias piezas sobre un tablero cuadrado, llamado "tablero de ajedrez". El jugador con las piezas blancas comienza la partida. Se dice que un jugador "está en juego" cuando se ha realizado la jugada de su adversario.
1.2.- El objetivo de cada jugador es situar al rey de su adversario "bajo ataque", de tal forma que el adversario no disponga de ninguna jugada legal. Del jugador que alcanza este objetivo se dice que ha dado "mate" al rey de su

adversario y que ha ganado la partida. No está permitido dejar el propio rey bajo ataque, ni exponerlo al ataque ni capturar el rey del oponente. El adversario, cuyo rey ha recibido el mate, pierde la partida.
1.3.- Si la posición es tal que ninguno de los jugadores puede dar mate, la partida es tablas.

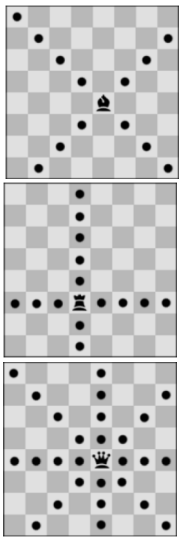
Artículo 2: La posición inicial de las piezas sobre el tablero

2.1.- El tablero de ajedrez es un cuadrado dividido en 64 casillas (escaques) cuadradas del mismo tamaño, con distribución 8 x 8, alternativamente claras (las casillas "blancas") y oscuras (las casillas "negras"). El tablero se coloca entre los jugadores de tal forma que la casilla de la esquina derecha más cercana a cada jugador sea blanca.
2.2.- Al comienzo de la partida, un jugador dispone de 16 piezas de color claro (las piezas "blancas"); el otro tiene 16 piezas de color oscuro (las piezas "negras"). Estas piezas son las siguientes:

- Un rey blanco, representado habitualmente por el símbolo: R
- Una dama blanca, representada habitualmente por el símbolo: D
- Dos torres blancas, representadas habitualmente por el símbolo: T
- Dos alfiles blancos, representados habitualmente por el símbolo: A
- Dos caballos blancos, representados habitualmente por el símbolo: C
- Ocho peones blancos, representados habitualmente por el símbolo: P
- Un rey negro, representado habitualmente por el símbolo: r
- Una dama negra, representada habitualmente por el símbolo: d
- Dos torres negras, representadas habitualmente por el símbolo: t
- Dos alfiles negros, representados habitualmente por el símbolo: a
- Dos caballos negros, representados habitualmente por el símbolo: c
- Ocho peones negros, representados habitualmente por el símbolo: p

2.3.- La posición inicial de las piezas sobre el tablero es la que se representa en la siguiente figura. A destacar, la dama de cada color está ubicada sobre un escaque del mismo color que la pieza.
2.4.- Las ocho hileras verticales de casillas se denominan "columnas". Las ocho hileras horizontales de casillas se denominan "filas". Una sucesión de casillas del mismo color en línea recta, tocándose por sus vértices, se denomina "diagonal".

Figura XXXIV: Posición inicial de piezas de ajedrez.



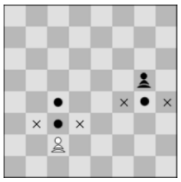
Artículo 3: El movimiento de las piezas

3.1.- No está permitido mover una pieza a una casilla ocupada por una pieza del mismo color. Si una pieza se mueve a una casilla ocupada por una pieza de su adversario, ésta es capturada y retirada del tablero como parte del mismo movimiento. Se dice que una pieza ataca a otra del adversario si puede efectuar una captura en esa casilla conforme a los Artículos 3.2 a 3.8. Se considera que una pieza ataca una casilla, incluso si no puede ser movida a dicha casilla porque este movimiento dejaría o colocaría su propio rey bajo ataque.
3.2.- El alfil puede ser movido a cualquier casilla a lo largo de una de las diagonales sobre las que se encuentra.
3.3.- La torre puede ser movida a cualquier casilla a lo largo de la fila o columna en las que se encuentra.

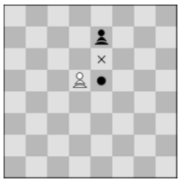
Figura XXXV: Movimientos de alfil, torre y dama.

3.4.- La dama puede ser movida a cualquier casilla a lo largo de la fila, columna o diagonal en las que se encuentra.
3.5.- Al realizar estos movimientos, el alfil, la torre o la dama no pueden pasar sobre ninguna otra pieza.

3.6.- El caballo puede ser movido a una de las casillas más próximas a la que se encuentra, sin ser de la misma fila, columna o diagonal.



3.7a.- El peón puede ser movido hacia adelante a la casilla inmediatamente delante suyo en la misma columna, siempre que dicha casilla esté desocupada; o
3.7a.- En su primer movimiento el peón puede ser movido como en (a); alternativamente, puede avanzar dos casillas a lo largo de la misma columna, siempre que ambas casillas estén desocupadas;
3.7c.- El peón puede ser movido a una casilla ocupada por una pieza del adversario que esté en diagonal delante suyo, sobre una columna adyacente, capturando dicha pieza.



3.7d.- Un peón que ataca una casilla atravesada por un peón del adversario que ha avanzado dos casillas en un movimiento desde su casilla original, puede capturarlo como si sólo hubiera avanzado una casilla. Esta captura sólo puede efectuarse en el movimiento inmediatamente siguiente al citado avance y se denomina captura "al paso".
3.7e.- Cuando un peón alcanza la fila más alejada desde su posición inicial debe ser cambiado, como parte del mismo movimiento, por una dama, torre, alfil o caballo del mismo color. La elección del jugador no está limitada a piezas que hayan sido capturadas anteriormente. Este cambio de un peón por otra pieza se denomina "promoción", siendo inmediato el efecto de la nueva pieza.

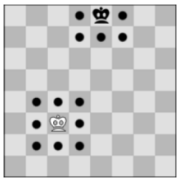
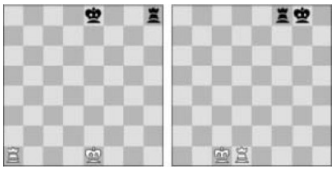


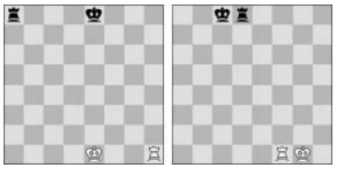
Figura XXXVI: Movimientos de caballo, peón y rey.

3.8.- Hay dos formas diferentes de mover el rey:

1.- desplazándolo a cualquier casilla adyacente no atacada por una o más piezas del adversario,
2.- "enrocando". El enroque es un movimiento del rey y de una de las torres del mismo color y que esté en la misma fila, que cuenta como una simple jugada del rey y que se realiza como sigue: el rey es trasladado dos casillas desde su casilla original hacia la torre y luego dicha torre es trasladada a la casilla que acaba de cruzar el rey.



Se ha perdido el derecho al enroque:
a). si el rey ya ha sido movido, o
b). con una torre que ya ha sido movida.



El enroque está temporalmente impedido:
a). si la casilla en la que se encuentra el rey, o la que debe cruzar, o la que finalmente va a ocupar, está atacada por una o más piezas del adversario,
b). si hay alguna pieza entre el rey y la torre con la que se va a efectuar el enroque.

Figura XXXVII: Enroques.

3.9.- Se dice que el rey está "en jaque" si está atacado por una o más piezas del adversario, incluso aunque dichas piezas no pudieran ser movidas porque dejarían o situarían a su propio rey en jaque. Ninguna pieza puede ser movida de forma que ponga o deje a su propio rey en jaque.

Artículo 4: La acción de mover las piezas

4.1.- Cada jugada debe efectuarse con una sola mano.
4.2.- El jugador que está en juego puede ajustar una o más piezas en sus casillas, siempre que previamente exprese su intención de hacerlo (por ejemplo, diciendo "compongo"). Exceptuando lo previsto en el Artículo 4.2, si el jugador que está en juego toca deliberadamente sobre el tablero:

a). una o más piezas propias, debe mover la primera pieza tocada que se pueda mover; o

b). una o más piezas del adversario, debe capturar la primera pieza tocada que pueda ser capturada; o

c). una pieza de cada color, debe capturar la pieza del adversario con la suya o, si ello es ilegal, mover o capturar la primera pieza tocada que se pueda mover o capturar. Si resulta imposible establecer qué pieza se tocó en primer lugar, será la pieza propia la que se considere como pieza tocada.

4.3a.- Si un jugador toca deliberadamente su rey y torre, debe enrocar por ese lado si fuera legal hacerlo.
4.3b.- Si un jugador toca deliberadamente una torre y luego su rey, no podrá enrocar por ese lado en esa jugada y se procederá según lo establecido en el Artículo 4.3a.
4.3c.- Si un jugador, con la intención de enrocar, toca el rey o rey y torre a la vez, siendo ilegal el enroque por ese lado, el jugador debe hacer otra jugada legal con su rey, lo que puede incluir el enroque por el otro lado. Si el rey no tiene ningún movimiento legal, el jugador es libre de realizar cualquier jugada legal.
4.3d.- Si un jugador promociona un peón, la elección de la pieza se considerará definitiva cuando ésta toque la casilla de promoción.
4.4.- Si ninguna de las piezas tocadas puede ser movida o capturada, el jugador puede realizar cualquier jugada legal.
4.5.- Cuando se ha dejado una pieza en una casilla, como jugada legal o parte de una jugada legal, ya no puede ser movida a otra casilla. Se considera que el movimiento ha sido realizado cuando se han cumplido todos los requisitos pertinentes del Artículo 3.

a). en el caso de una captura, cuando la pieza capturada se ha retirado del tablero y el jugador ha soltado su propia pieza después de situarla en su nueva casilla;

b). en el caso del enroque, cuando el jugador ha soltado la torre en la casilla previamente cruzada por el rey. Cuando el jugador ha soltado el rey la jugada no se considera todavía completada, pero el jugador ya no tiene otra opción que realizar el enroque por ese lado, siempre que sea legal;

c). en el caso de la promoción de un peón, cuando éste ha sido retirado del tablero y la mano del jugador ha soltado la nueva pieza después de situarla en la casilla de promoción. Cuando el jugador ha soltado el peón que ha alcanzado la casilla de promoción, la jugada no se considera todavía completada, pero el jugador no tiene ya el derecho de jugar el peón a otra casilla.

4.7.- Un jugador pierde su derecho a reclamar la violación por parte de su adversario de los artículos 4.3 o 4.4 una vez que él mismo haya tocado deliberadamente una pieza.

Artículo 5: La finalización de la partida

5.1.-

a). La partida es ganada por el jugador que ha dado mate al rey de su adversario. Esto finaliza inmediatamente la partida, siempre que la jugada que generó la posición de mate fuera una jugada legal.

b). La partida es ganada por el jugador cuyo adversario declara que abandona. Esto finaliza inmediatamente la partida.

5.2.-

a). La partida es tablas cuando el jugador que está en juego no puede hacer ninguna jugada legal y su rey no está en jaque. Se dice entonces que el rey está "ahogado". Esto finaliza inmediatamente la partida, siempre que la jugada que generó la posición de rey ahogado fuera legal.

b). La partida es tablas cuando se alcanza una posición en la que ningún jugador puede dar mate al rey del adversario con cualquier serie de jugadas legales. Se dice entonces que la partida termina en una "posición muerta". Esto finaliza inmediatamente la partida, siempre que la jugada que generó tal posición fuera legal.

c). La partida es tablas por acuerdo entre los dos jugadores durante el desarrollo de la misma. Esto finaliza inmediatamente la partida. (Ver Artículo 9.1).

d). La partida puede ser tablas si se va a dar o ya se ha dado cualquier posición idéntica al menos tres veces sobre el tablero. (Ver Artículo 9.2).

e). La partida puede ser tablas si cada jugador ha hecho los últimos 50 movimientos consecutivos sin que haya habido ningún movimiento de peón ni captura de pieza. (Ver Artículo 9.3).

Artículo 6: El reloj de ajedrez

6.1.- Se entiende por "reloj de ajedrez" un dispositivo con dos indicadores de tiempo, conectados entre sí de tal forma que sólo uno de ellos pueda funcionar en cada momento. En las presentes Leyes, el término "reloj" hace referencia a uno de los dos citados indicadores de tiempo. Se entiende por "caída de bandera" la finalización del tiempo asignado a un jugador.

6.2.-

a). Al utilizar un reloj de ajedrez, cada jugador debe realizar un número mínimo de jugadas, o todas, en un período de tiempo prefijado y/o puede recibir un determinado tiempo adicional con cada jugada. Todas estas cuestiones deben ser especificadas con antelación.

b). El tiempo no consumido por un jugador durante un período se añade a su tiempo disponible para el siguiente período, excepto en la modalidad de "demora". En la modalidad de demora, ambos jugadores reciben un determinado "tiempo principal de reflexión". Además, por cada jugada reciben un "tiempo extra fijo". El descuento del tiempo principal comienza sólo cuando el tiempo extra se ha agotado. El tiempo principal de reflexión no cambia siempre que el jugador detenga su reloj antes de que se agote el tiempo extra, independientemente de la proporción de tiempo extra utilizado.

6.3.- Cada reloj dispone de una "bandera". Inmediatamente después de una caída de bandera, deben comprobarse los requisitos del Artículo 6.2(a).

6.4.- Antes del comienzo de la partida, el árbitro decide la ubicación del reloj de ajedrez.

6.5.- A la hora especificada para el comienzo de la partida, se pone en marcha el reloj del jugador que tiene las piezas blancas.

6.6.- Si al comienzo de la partida no está presente ninguno de los jugadores, el jugador de blancas perderá todo el tiempo transcurrido hasta su llegada, salvo que las bases de la competición o el árbitro decidan otra cosa.

6.7.- Cualquier jugador que se presente ante el tablero con más de una hora de retraso sobre la hora programada para el comienzo de la sesión, perderá la partida salvo que las bases de la competición o el árbitro decidan otra cosa.

6.8.-

a). Durante la partida, cada jugador, una vez realizada su jugada sobre el tablero, detendrá su reloj y pondrá en marcha el de su adversario. A un jugador siempre se le debe permitir detener su reloj. Hasta que el jugador lo haya hecho así, no se considera que su jugada esté completada, salvo que la jugada que ha realizado finalice la partida (ver Artículos 5.1 y 5.2). El tiempo transcurrido entre realizar la jugada sobre el tablero y detener su reloj, poniendo en marcha el del adversario, se considera como parte del tiempo asignado al jugador.

b). Un jugador debe detener su reloj con la misma mano con la que realizó su jugada. Está prohibido que un jugador mantenga el dedo sobre el pulsador o rondando por encima del mismo.

c). Los jugadores deben manejar el reloj de ajedrez correctamente. Está prohibido golpearlo violentamente, cogerlo o tirarlo. Un manejo incorrecto del reloj será penalizado de acuerdo con el Artículo 13.4.

d). Si un jugador está imposibilitado para utilizar el reloj, puede ser acompañado por un asistente, bajo la aceptación del árbitro, para desempeñar dicha labor. El árbitro ajustará su reloj de una forma proporcionada.

6.9.- Se considera que una bandera ha caído cuando el árbitro lo observa o cuando uno de los dos jugadores ha efectuado una reclamación válida en ese sentido.

6.10.- Excepto los casos donde se apliquen los Artículos 5.1, 5.2(a), 5.2(b) o 5.2(c), si un jugador no completa el número prescrito de jugadas en el tiempo asignado, pierde la partida. Sin embargo, la partida es tablas si la posición es tal que el adversario no puede dar mate al jugador mediante cualquier posible combinación de jugadas legales, incluso jugando de la forma más torpe.

6.11.- Toda indicación proporcionada por los relojes se considera concluyente en ausencia de algún defecto evidente. Un reloj de ajedrez con un defecto evidente debe ser reemplazado. El árbitro hará uso de su mejor criterio al determinar los tiempos que deben aparecer en el reloj de ajedrez reemplazado.

6.12.- Si han caído ambas banderas y es imposible establecer cuál de ellas lo hizo en primer lugar, entonces:

a). continuará la partida si se trata de cualquier periodo excepto el último.

b). la partida será declarada tablas si se trata del último periodo, en el que se han de realizar todas las jugadas restantes.

6.13.-

a). Si la partida debe ser interrumpida, el árbitro detendrá los relojes.

b). Un jugador puede detener los relojes sólo para solicitar la asistencia del árbitro, por ejemplo cuando se ha producido una promoción y no está disponible la pieza requerida.

c). En cualquier caso, el árbitro decidirá cuándo se va a reanudar la partida.

d). Si un jugador detiene los relojes para solicitar la asistencia del árbitro, éste determinará si el jugador tenía una razón válida para hacerlo. Si resulta obvio que el jugador no tenía una razón válida para detener los relojes, el jugador será sancionado conforme al Artículo 13.4.

6.14.- Si se produce una irregularidad y/o las piezas deben ser reubicadas a una posición anterior, el árbitro hará uso de su mejor criterio para determinar los tiempos que deben aparecer en los relojes. Si fuera necesario, deberá ajustar también el contador de jugadas del reloj.

6.15.- En el recinto de juego se permite el uso de pantallas, monitores o tableros murales que muestren la posición actual sobre el tablero, los movimientos y el número de jugadas realizadas, así como relojes que muestren incluso el número de jugadas realizadas. Sin embargo, el jugador no puede realizar una reclamación basándose en la información mostrada por este tipo de dispositivos.

Artículo 7: Irregularidades

7.1.-

a). Si en el curso de una partida se comprueba que la posición inicial de las piezas era incorrecta, la partida será anulada y se jugará una nueva.

b). Si en el curso de una partida se comprueba que el único error ha sido que el tablero no se colocó de acuerdo con el Articulo 2.1, la partida continuará pero la posición deberá transferirse a un tablero colocado correctamente.

7.2.- Si una partida ha comenzado con los colores invertidos, la misma continuará salvo que el árbitro decida otra cosa.

7.3.- Si un jugador desplaza una o más piezas, restablecerá la posición correcta en su propio tiempo. Si fuera necesario, tanto el jugador como su adversario pueden detener los relojes y solicitar la asistencia del árbitro. Éste puede sancionar al jugador que desplazó las piezas.

7.4.-

a). Si en el curso de una partida se comprueba que se ha realizado una jugada ilegal, incluyendo el incumplimiento de los requisitos relativos a la promoción de un peón o la captura del rey del adversario, se restablecerá la posición previa a producirse la irregularidad. Si no puede determinarse la posición inmediata anterior a producirse la irregularidad, la partida continuará a partir de la última posición identificable previa a la irregularidad. Los relojes se ajustarán de acuerdo con el Artículo 6.14. A la jugada que reemplace a la jugada ilegal se le aplica el Artículo 4.3. Después continuará la partida a partir de la antes citada posición restablecida.

b). Después de actuar conforme al Artículo 7.4(a), para las dos primeras jugadas ilegales el árbitro concederá en cada caso dos minutos de tiempo extra al adversario; para una tercera jugada ilegal del mismo jugador, el árbitro decretará la pérdida de la partida para el infractor.

Si durante una partida se comprueba que algunas piezas han sido desplazadas de sus casillas, deberá restablecerse la posición anterior a la irregularidad. Si no puede identificarse dicha posición, la partida continuará a partir de la última posición identificable anterior a que se produjera la irregularidad. Los relojes serán ajustados de acuerdo al Artículo 6.14. Después continuará la partida a partir de la antes citada posición restablecida.

Artículo 8: La anotación de las jugadas

8.1.- En el transcurso de la partida cada jugador está obligado a anotar sus propias jugadas y las de su adversario en la forma correcta, jugada tras jugada, de forma tan clara y legible como sea posible, en anotación algebraica, en la planilla prescrita para la competición. Está prohibido anotar las jugadas antes de realizarlas, excepto cuando el jugador esté reclamando tablas según los artículos 9.2 o 9.3. Si así lo desea, un jugador puede replicar a una jugada de su adversario antes de anotarla. Debe anotar su jugada previa antes de realizar otra. Ambos jugadores deben anotar en su planilla la oferta de tablas. Si un jugador está imposibilitado para anotar, puede ser acompañado por un asistente, bajo la aceptación del árbitro, para desempeñar dicha labor. El árbitro ajustará su reloj de una forma proporcionada.

8.2.- La planilla estará a la vista del árbitro durante toda la partida.

8.3.- Las planillas son propiedad de los organizadores del torneo.

8.4.- Si un jugador dispone de menos de cinco minutos en su reloj y no tiene un tiempo adicional de 30 segundos o más añadidos con cada jugada, no está obligado a cumplir los requisitos del Artículo 8.1. Inmediatamente después de que haya caído una bandera, el jugador debe actualizar completamente su planilla antes de mover una pieza sobre el tablero.

8.5.-

a). Si ningún jugador está obligado a anotar la partida según el Artículo 8.4, el árbitro o un asistente procurará estar presente y anotar las jugadas. En tal caso, inmediatamente después de que haya caído una bandera, el árbitro parará los relojes. Después ambos jugadores actualizarán sus planillas, utilizando la planilla del árbitro o la del adversario.

b). Si sólo un jugador no está obligado a anotar la partida según el Artículo 8.4, debe actualizar completamente su planilla en cuanto haya caído cualquiera de las dos banderas antes de mover cualquier pieza en el tablero. Supuesto que el jugador esté en juego, puede utilizar la planilla de su adversario pero tendrá que devolverla antes de hacer una jugada.

c). Si no hay ninguna planilla completa disponible, los jugadores deben reconstruir la partida sobre un segundo tablero bajo el control del árbitro o un asistente. Primero, y antes de que se realice la reconstrucción, el árbitro anotará la posición actual de la partida, el tiempo transcurrido en ambos relojes y el número de jugadas realizadas, si se dispone de tal información.

8.6.- Si no se pueden actualizar las planillas de forma que muestren que un jugador ha sobrepasado el tiempo asignado, la siguiente jugada realizada será considerada como la primera del siguiente periodo de tiempo, salvo que sea evidente que se han realizado más jugadas.

8.7.- Al finalizar la partida ambos jugadores firmarán las dos planillas, indicando el resultado. Dicho resultado permanecerá aunque sea incorrecto, salvo que el árbitro decida otra cosa.

Artículo 9: La partida tablas

9.1.-

a). Un jugador que desee ofrecer tablas deberá hacerlo después de haber hecho una jugada sobre el tablero y antes de detener su reloj y poner en marcha el del adversario. Una oferta en cualquier otro momento de la partida será válida, pero se tendrá en consideración el Artículo 12.6. La oferta no puede ir ligada a ninguna condición. En ambos casos, no se puede retirar la oferta y mantiene su validez hasta que el adversario la acepte, la rechace bien verbalmente o bien tocando una pieza con intención de moverla o capturarla, o la partida concluya de alguna otra forma.

b). La oferta de tablas será anotada por cada jugador en su planilla con un símbolo. (ver Apéndice E13).

c). Una reclamación de tablas conforme a los Artículos 9.2, 9.3 ó 10.2 tendrá la consideración de una oferta de tablas.

9.2.- La partida es tablas, bajo una correcta reclamación del jugador que está en juego, cuando la misma posición, al menos por tercera vez (no necesariamente por repetición secuencial de jugadas):

a). va a producirse, si el jugador primero anota su jugada en su planilla y declara al árbitro su intención de realizarla; o

b). acaba de producirse, añadiéndose la circunstancia de que el reclamante está en juego.

Se considera que las posiciones como las de (a) y (b) son la misma si es el mismo jugador quien está en juego, las piezas del mismo tipo y color ocupan las mismas casillas, y los movimientos posibles de todas las piezas de ambos jugadores son los mismos.

Las posiciones no se consideran la misma si un peón pudiera haber sido capturado al paso -ya que después no podrá ser capturado- o si ha cambiado el derecho a enrocar, temporalmente o de forma permanente.

9.3.- La partida es tablas, bajo una correcta reclamación del jugador que está en juego, si:

a). escribe en su planilla y declara al árbitro su intención de realizar una jugada que dará lugar a que, en los últimos 50 movimientos consecutivos por cada jugador, no se haya movido ningún peón ni se haya capturado alguna pieza; o

b). se hayan producido los últimas 50 movimientos consecutivos de cada jugador sin moverse algún peón y sin realizarse ninguna captura.

9.4.- Si un jugador realiza una jugada sin haber reclamado tablas, pierde el derecho a reclamarlas en esa jugada, en base a los Artículos 9.2 y 9.3.

9.5.- Si un jugador reclama tablas en base a los Artículos 9.2 ó 9.3, parará inmediatamente ambos relojes. No se le permite retirar su reclamación.

a). Si se comprueba que la reclamación es correcta, la partida es tablas de forma inmediata.

b). Si se comprueba que la reclamación es incorrecta, el árbitro añadirá tres minutos al tiempo restante de su adversario. Además, si el reclamante dispone de más de dos minutos en su reloj, el árbitro deducirá la mitad de dicho tiempo hasta un máximo de tres minutos. Si al reclamante le quedara más de un minuto pero menos de dos, su tiempo

restante será de un minuto. Si al reclamante le quedara menos de un minuto, el árbitro no realizará ajustes en su reloj. Luego la partida continuará y deberá hacerse la jugada anunciada.
9.6.- La partida es tablas cuando se llega a una posición a partir de la cual no puede producirse un mate mediante ninguna posible combinación de jugadas legales, incluso jugando de la forma más torpe. Esto finaliza inmediatamente la partida, siempre que la jugada que produjo esta posición fuera legal.

Artículo 10: Final a caída de bandera

10.1.- Un "final a caída de bandera" es la última fase de una partida, cuando todas las jugadas (restantes) deben hacerse en un tiempo limitado.
10.2.- Si al jugador que está en juego le quedan menos de dos minutos en su reloj, puede reclamar tablas antes de que caiga su bandera. Parará los relojes y requerirá la presencia del árbitro.

a). Si el árbitro está de acuerdo en que el adversario no está haciendo ningún esfuerzo para ganar la partida por procedimientos normales, o que no es posible ganar por procedimientos normales, declarará la partida tablas. En otro caso aplazará su decisión o rechazará la reclamación.

b). Si el árbitro aplaza su decisión, puede otorgar al adversario dos minutos de tiempo extra de reflexión y la partida continuará, si ello fuera posible, en presencia de un árbitro. El árbitro decretará el resultado final posteriormente, durante la partida o después de que una bandera haya caído. Declarará la partida tablas si está de acuerdo en que la posición final no puede ser ganada por medios normales, o que el adversario no estaba haciendo lo suficiente para ganar por medios normales.

c). Si el árbitro ha rechazado la reclamación, se le concederán dos minutos de tiempo extra de reflexión al adversario.

d). En relación con los anteriores 10.2(a), (b) o (c), la decisión del árbitro será definitiva.

Artículo 11: Puntuación

11.1.- Salvo que se haya anunciado otra cosa de antemano, un jugador que gana su partida, o que gana por incomparecencia, recibe un punto (1); un jugador que pierde su partida, o que pierde por incomparecencia, recibe cero puntos (0); y un jugador que entabla su partida recibe medio punto (½).

Artículo 12: La conducta de los jugadores

12.1.- Los jugadores no actuarán de forma que deshonren el juego del ajedrez.
12.2.-

a). Durante la partida está prohibido que los jugadores hagan uso de cualquier tipo de notas, fuentes de información, consejos o análisis en otro tablero.

b). Está estrictamente prohibido llevar teléfonos móviles o cualquier otro medio electrónico de comunicación, no permitido por el árbitro, dentro del recinto de juego. Si durante las partidas suena en el recinto de juego el móvil de un jugador, éste perderá su partida. Será el árbitro quien determine la puntuación del adversario.
12.3.- La planilla sólo se utilizará para anotar las jugadas, el tiempo indicado en los relojes, la oferta de tablas, cuestiones relativas a una reclamación y otros datos relevantes.
12.4.- Los jugadores que hayan finalizado sus partidas serán considerados espectadores.
12.5.- No está permitido a los jugadores abandonar el recinto de juego sin permiso del árbitro. Se entiende por "recinto de juego" el constituido por las zonas de juego, de descanso, de fumadores, aseos y cualesquiera otras designadas por el árbitro.
Al jugador que está en juego no le está permitido abandonar la zona de juego sin permiso del árbitro.
12.6.- Está prohibido distraer o molestar al adversario de cualquier manera. Esto incluye reclamaciones u ofertas de tablas improcedentes.
12.7.- Las infracciones a las reglas establecidas en los Artículos 12.1 a 12.6 darán lugar a sanciones conforme al Artículo 13.4.
12.8.- La persistente negativa de un jugador a cumplir con las Leyes del Ajedrez será sancionada con la pérdida de la partida. El árbitro decidirá la puntuación del adversario.
12.9.- Si ambos jugadores son considerados culpables de acuerdo con el Artículo 12.8, la partida se declarará perdida para ambos.

Artículo 13: El papel del árbitro (ver el Prólogo)

13.1.- El árbitro se cuidará de que se cumplan estrictamente las Leyes del Ajedrez.
13.2.- El árbitro actuará en el mejor provecho de la competición. Se asegurará de que se mantengan unas buenas condiciones de juego y de que los jugadores no sean molestados. Supervisará el desarrollo de la competición.
13.3.- El árbitro observará las partidas, especialmente cuando los jugadores estén apurados de tiempo, exigirá el cumplimiento de las decisiones que haya adoptado y, cuando corresponda, impondrá sanciones a los jugadores.
13.4.- El árbitro puede aplicar una o más de las siguientes sanciones:

a). una advertencia;
b). aumentar el tiempo de reflexión que disponga el adversario;
c). reducir el tiempo de reflexión que disponga el infractor;
d). decretar la pérdida de la partida;
e). reducir la puntuación obtenida en una partida al infractor;
f). aumentar la puntuación obtenida en una partida por el adversario, hasta el máximo posible para dicha partida;
g). la expulsión de la competición.

13.5.- El árbitro puede conceder tiempo adicional a uno o ambos jugadores en caso de desorden ajeno a la partida.
13.6.- El árbitro no puede intervenir en una partida excepto en los casos descritos en las Leyes del Ajedrez. No indicará el número de jugadas realizadas, excepto en aplicación del Artículo 8.5 cuando al menos una de las banderas haya caído. El árbitro se abstendrá de informar a un jugador de que su adversario ha realizado una jugada o de que no ha pulsado su reloj.
13.7.-

a). Los espectadores y jugadores de otras partidas no pueden hablar o interferir de cualquier otro modo en una partida. Si fuera necesario, el árbitro puede expulsar del recinto de juego a los infractores.
b). Está prohibido para cualquiera el uso de teléfonos móviles en el recinto de juego y en cualquier otra zona designada por el árbitro.

Artículo 14: FIDE

14.1.- Las federaciones afiliadas pueden solicitar a la FIDE que adopte una decisión oficial en problemas relacionados con las Leyes del Ajedrez.

APÉNDICES

A Partidas aplazadas

A1.-

a). Si una partida no ha terminado al final del tiempo establecido para la sesión, el árbitro exigirá al jugador que esté en juego que "selle" su jugada. El jugador debe anotar su jugada de forma clara e inequívoca en su planilla, poner su planilla y la de su adversario en un sobre, cerrar el sobre y, sólo entonces, parar su reloj sin poner en marcha el de su adversario. Hasta que haya parado los relojes, el jugador mantiene el derecho a cambiar su jugada secreta. Si, tras haber sido advertido por el árbitro para que selle su jugada, el jugador realiza una jugada sobre el tablero, debe anotar dicha jugada en su planilla como su jugada secreta.
b). Un jugador que, estando en juego, aplaza la partida antes del fin de la sesión de juego, se considerará que ha realizado la jugada secreta a la hora establecida para el fin de la sesión, indicándose así su tiempo restante.
A2.- En el sobre se escribirán los siguientes datos:

a). los nombres de los jugadores;
b). la posición inmediatamente anterior a la jugada secreta;
c). el tiempo empleado por cada jugador;
d). el nombre del jugador que ha realizado la jugada secreta;
e). el número de la jugada secreta;
f). la oferta de tablas, si la propuesta sigue vigente;
g). la fecha, hora y lugar de reanudación de la partida.

A3.- El árbitro comprobará la exactitud de la información escrita en el sobre y es el responsable de su custodia.
A4.- Si un jugador propone tablas después de que su adversario haya sellado su jugada, la oferta permanece válida hasta que el adversario la haya aceptado o rechazado, como en el Artículo 9.1.

A5.- Antes de reanudarse la partida, se pondrá sobre el tablero la posición inmediatamente anterior a la jugada secreta y se indicará en los relojes el tiempo empleado por cada jugador en el momento del aplazamiento.
A6.- Si antes de la reanudación de la partida se acuerdan tablas o uno de los jugadores notifica al árbitro que abandona, finaliza la partida.
A7.- El sobre se abrirá sólo cuando esté presente el jugador que debe responder a la jugada secreta.
A8.- Excepto en los casos mencionados en los Artículos 6.10 y 9.6, pierde la partida un jugador cuya anotación de la jugada secreta:

a). sea ambigua; o
b). esté escrita de tal forma que sea imposible establecer su significado real; o bien
c). resulte ser una jugada ilegal.

A9.- Si a la hora acordada para la reanudación:

a). Está presente el jugador que tiene que responder a la jugada secreta, se abre el sobre, se ejecuta la jugada secreta sobre el tablero y se pone en marcha su reloj.

b). No está presente el jugador que tiene que responder a la jugada secreta, se pondrá en marcha su reloj. Al llegar, el jugador puede detener su reloj y requerir la presencia del árbitro. Es entonces cuando se abre el sobre y se ejecuta la jugada secreta sobre el tablero, volviéndose a poner en marcha su reloj.

c). No está presente el jugador que selló su jugada, su adversario tiene derecho a anotar su respuesta en la planilla, sellar su jugada en un nuevo sobre, detener su reloj y poner en marcha el del jugador ausente, en lugar de ejecutar su jugada en la forma normal. Si así fuere, el sobre se lo pasará al árbitro para su custodia y se abrirá a la llegada del jugador ausente.

A10.- Perderá la partida el jugador que se presente ante el tablero con más de una hora de retraso sobre la hora programada para la reanudación de una partida aplazada (salvo que las bases de la competición o el árbitro decidan otra cosa).
No obstante, si el ausente es el jugador que hizo la jugada secreta, la partida se decide de otra forma, si:

a). el jugador ausente ha ganado la partida porque la jugada secreta da mate; o

b). el jugador ausente ha entablado la partida porque la jugada secreta genera una posición de rey ahogado o una posición como la descrita en el Artículo 9.6; o bien

c). el jugador presente ante el tablero ha perdido la partida conforme al Artículo 6.10.

A11.-

a). Si se pierde el sobre que contiene la jugada secreta, la partida continuará a partir de la posición y con los tiempos anotados en el momento del aplazamiento. Si no se pudiera restablecer el tiempo consumido por cada jugador, el árbitro ajustará los relojes. El jugador que hizo la jugada secreta, ejecuta sobre el tablero la jugada que declara ser la que selló.

b). Si es imposible restablecer la posición, se anula la partida y deberá jugarse una nueva.

A12.- Si, en la reanudación de la partida y antes de hacer su primera jugada, uno de los jugadores muestra que el tiempo consumido ha sido puesto incorrectamente en alguno de los relojes, el error debe ser corregido. Si no se prueba el error en ese momento, la partida continúa sin hacer correcciones, salvo que el árbitro considere que las consecuencias serían demasiado graves.
A13.- La duración de cada sesión de reanudación estará controlada por el reloj del árbitro. Las horas de comienzo y finalización serán anunciadas con antelación.

B Ajedrez Rápido

B1.- Una "partida de Ajedrez Rápido" es aquella en la que todas las jugadas deben efectuarse en un tiempo fijo de 15 a 60 minutos para cada jugador; o cuando el resultado de la suma del tiempo asignado más el incremento por jugada multiplicado por 60 está incluido entre 15 y 60 minutos.
B2.- El juego se regirá por las Leyes del Ajedrez de la FIDE, excepto en lo que quede anulado por las siguientes reglas de Ajedrez Rápido.
B3.- Los jugadores no están obligados a anotar las jugadas.
B4.- Una vez completadas tres jugadas por cada jugador, no pueden hacerse reclamaciones relativas a una incorrecta ubicación de las piezas, la colocación del tablero o los tiempos que figuran en los relojes. En caso de emplazamiento cambiado de rey y dama, no se permite el enroque con ese rey.
B5.- El árbitro adoptará una decisión con respecto al Artículo 4 (pieza tocada) sólo a requerimiento de uno o de ambos jugadores.
B6.- Se considera que se ha completado una jugada ilegal cuando se ha puesto en marcha el reloj del adversario. El adversario cuenta entonces con el derecho, siempre antes de realizar su jugada, de reclamar que el jugador ha

completado una jugada ilegal. Únicamente tras dicha reclamación adoptará el árbitro una decisión. En cambio, el árbitro intervendrá, si es posible, cuando ambos reyes estén en jaque o cuando no se haya completado la promoción de un peón.
B7.- Se considera que la bandera ha caído cuando un jugador ha hecho una reclamación válida a tal efecto. El árbitro se abstendrá de señalar una caída de bandera.
B8.- Para reclamar una victoria por tiempo, el reclamante debe detener ambos relojes y notificárselo al árbitro. Para que la reclamación prospere, la bandera del reclamante debe permanecer en alto, y la de su adversario caída, después de que se hayan parado los relojes.
B9.- Si ambas banderas han caído, la partida es tablas.

C Ajedrez Relámpago

C1.- Una "partida de Ajedrez Relámpago" es aquella en la que todas las jugadas deben efectuarse en un tiempo fijo menor a 15 minutos para cada jugador; o cuando el resultado de la suma del tiempo asignado más el incremento por jugada multiplicado por 60 sea menor de 15 minutos.
C2.- El juego se regirá por las Leyes del Ajedrez Rápido, como en el Apéndice B, excepto en lo que quede anulado por las siguientes reglas de Ajedrez Relámpago. Los artículos 10.2 y B.6 no son aplicables.
C3.- Una jugada ilegal se completa en cuanto se pone en marcha el reloj del adversario. En cualquier caso, el adversario está autorizado a reclamar la victoria antes de realizar su propia jugada. Si el adversario no puede alcanzar el mate por cualquier serie de jugadas legales, incluso con las respuestas más torpes, entonces está autorizado a reclamar tablas antes de realizar su propia jugada. Una vez que el adversario ha realizado su jugada, no puede corregirse una jugada ilegal.

D Finales a caída de bandera cuando no hay un árbitro en el recinto de juego

D1.- Cuando se disputan partidas de acuerdo con el Artículo 10, un jugador puede reclamar tablas cuando le queden menos de dos minutos en su reloj y antes de que caiga su bandera. Esto finaliza la partida. Puede reclamar en base a que:

a). su adversario no puede ganar por procedimientos normales, o
b). su adversario no ha estado haciendo ningún esfuerzo para ganar por procedimientos normales.

En (a) el jugador debe anotar la posición final y su adversario verificarla.
En (b) el jugador debe anotar la posición final y presentar como prueba una planilla actualizada. El adversario verificará tanto la planilla como la posición final. La reclamación será remitida a un árbitro, cuya decisión será la definitiva.

E Notación Algebraica

La FIDE sólo reconoce para sus propios torneos y matches un sistema de anotación, el Sistema Algebraico, y recomienda también el uso de este sistema uniforme de anotación ajedrecística para literatura y revistas de ajedrez. Las planillas en las que se utilice un sistema de anotación distinto al algebraico, no podrán ser utilizadas como prueba en casos en los que la planilla de un jugador se usa para tal fin. Un árbitro que observe que un jugador utiliza un sistema de anotación distinto al algebraico, advertirá al jugador acerca de este requisito.

Descripción del Sistema Algebraico

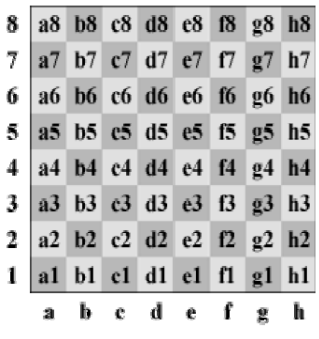


Figura XXXVIII: Notación de escaques.

E1.- En esta descripción, pieza significa cualquier pieza excepto un peón.
E2.- Cada pieza es indicada por la primera letra, en mayúscula, de su nombre. Por ejemplo: R = rey, D = dama, T = torre, A = alfil, C = caballo.
E3.- Para la letra inicial del nombre de una pieza, cada jugador es libre de usar la primera letra del nombre utilizado normalmente en su país. Por ejemplo: A = alfil (en español); F = fou (en francés); L = loper (en alemán). En publicaciones impresas se recomienda el uso de figuras.
E4.- Los peones no son indicados por su primera letra, sino que se les reconoce precisamente por la ausencia de la misma. Por ejemplo: e5, d4, a5.
E5.- Las ocho columnas (de izquierda a derecha para las blancas y de derecha a izquierda para las negras) son indicadas por las letras minúsculas: a, b, c, d, e, f, g y h, respectivamente.

E6.- Las ocho filas (de abajo arriba para las blancas y de arriba abajo para las negras) están numeradas: 1, 2, 3, 4, 5, 6, 7 y 8, respectivamente. Consecuentemente, en la posición inicial las piezas y peones blancos se colocan en la primera y segunda filas; las piezas y peones negros en la octava y séptima filas.
E7.- Como consecuencia de las reglas anteriores, cada una de las 64 casillas está indicada invariablemente por una sola combinación de una letra y un número.

E8.- Cada movimiento de una pieza se indica por (a) la letra inicial del nombre de la pieza en cuestión y (b) la casilla de llegada. No hay guión entre (a) y (b). Por ejemplo: Ae5, Cf3, Td1. En el caso de peones, sólo se indica la casilla de llegada. Por ejemplo: e5, d4, a5.
E9.- Cuando una pieza realiza una captura, se inserta una x entre (a) la letra inicial del nombre de la pieza en cuestión y (b) la casilla de llegada. Por ejemplo: Axe5, Cxf3, Txd1. Cuando un peón realiza una captura, debe indicarse la columna de partida, luego una x y finalmente la casilla de llegada. Por ejemplo: dxe5, gxf3, axb5. En el caso de una captura al paso, se pone como casilla de llegada la que finalmente ocupa el peón que realiza la captura, añadiéndose a la anotación "a.p.". Ejemplo: exd6 a.p..
E10.- Si dos piezas idénticas pueden moverse a la misma casilla, la pieza que se mueve se indica como sigue:

1). Si ambas piezas están en la misma fila: por (a) la letra inicial del nombre de la pieza, (b) la columna de la casilla de salida y (c) la casilla de llegada.

2). Si ambas piezas están en la misma columna: por (a) la letra inicial del nombre de la pieza, (b) la fila de la casilla de salida y (c) la casilla de llegada.

3). Si las piezas están en filas y columnas distintas, es preferible el método (1).

En el caso de una captura, debe insertarse una x entre (b) y (c). Ejemplos:
Hay dos caballos, en las casillas g1 y e1, y uno de ellos mueve a la casilla f3: según el caso, puede ser Cgf3 o Cef3.
Hay dos caballos, en las casillas g5 y g1, y uno de ellos mueve a la casilla f3: según el caso, puede ser C5f3 o C1f3.
Hay dos caballos, en las casillas h2 y d4, y uno de ellos mueve a la casilla f3: según el caso, puede ser Chf3 o Cdf3.
Si se da una captura en la casilla f3, se modifican los ejemplos anteriores insertando una x: según el caso, puede ser (1) Cgxf3 o Cexf3, (2) C5xf3 o C1xf3, (3) Chxf3 o Cdxf3.
E11.- Si dos peones pueden capturar la misma pieza o peón del adversario, el peón que se mueve se indica por (a) la letra de la columna de salida, (b) una x, (c) la casilla de llegada. Por ejemplo: si hay peones blancos en las casillas c4 y e4 y un peón o pieza negro en la casilla d5, la anotación para la jugada de Blancas puede ser, según el caso: cxd5 o exd5.
E12.- En el caso de la promoción de un peón, se indica el movimiento efectivo del peón, seguido inmediatamente por la letra inicial de la nueva pieza. Por ejemplo: d8D, f8C, b1A, g1T.
E13.- La oferta de tablas se anotará como (=).
E14.- Abreviaturas esenciales:

0-0	enroque con la torre de h1 o de h8 (enroque por el flanco de rey o enroque corto).
0-0-0	enroque con la torre de a1 o de a8 (enroque por el flanco de dama o enroque largo).
X	captura.
+	Jaque.
++ o #	mate.
a.p.	captura de peón al paso.

Ejemplo de partida: 1e4 e5 2Cf3 Cf6 3.d4 exd4 4.e5 Ce4 5Dxd5 d5 6.exd6 a.p. Cxd6 7Ag5 Cc6 8.De3+ Ae7 9.Cbd2 0-0 10.0-0-0 Te8 11.Rb1(=)

F - Reglas para partidas con ciegos y discapacitados visuales

F1.- La organización del torneo tendrá la potestad de adaptar las siguientes reglas conforme a las circunstancias concretas. En el ajedrez de competición, cuando uno de los jugadores es discapacitado visual (legalmente ciego), cualquiera de los contendientes puede exigir la utilización de dos tableros, utilizando el jugador discapacitado visual un tablero especialmente adaptado y su adversario uno normal.
El tablero especial debe reunir los siguientes requisitos:

a). un tamaño de al menos 20 x 20 centímetros;
b). las casillas negras ligeramente elevadas;
c). un agujero de seguridad en cada casilla;
d). cada pieza provista de una clavija que encaje en el agujero de seguridad;
e). piezas de diseño Staunton, con las piezas negras especialmente identificadas.

F2.- El juego se regirá por las siguientes reglas:

1. Las jugadas se anunciarán claramente, repetidas por el adversario y ejecutadas en su tablero. Para hacer que tal anuncio sea lo más claro posible, se sugiere la utilización de los siguientes nombres en lugar de las correspondientes letras del sistema algebraico:

a) - Anna
b) - Bella
c) - Cesar
d) - David
e) - Eva
f) - Felix
g) - Gustav
h) - Hector

Las filas, de blancas a negras, recibirán los números en alemán:

1) - eins
2) - zwei
3) - drei
4) - vier
5) - fuenf
6) - sechs
7) - sieben
8) - acht

El enroque se anuncia “Lange Rochade” (en alemán para el enroque largo) y “Kurze Rochade” (en alemán para el enroque corto).

Las piezas llevan los nombres: Koenig (rey), Dame (dama), Turm (torre), Laeufer (alfil), Springer (caballo) y Bauer (peón).

2. En el tablero del jugador discapacitado visual, se considerará “tocada” una pieza cuando haya sido sacada del agujero de seguridad.

3. Se considerará una jugada “realizada” cuando:

a). en el caso de una captura, la pieza capturada ha sido retirada del tablero del jugador al que le corresponde jugar;

b). se coloca una pieza en un agujero de seguridad distinto;

c). la jugada ha sido anunciada.

Sólo entonces se pondrá en marcha el reloj del adversario.

En lo que se refiere a estas reglas 2 y 3 (consideración de pieza tocada y jugada realizada), para el jugador no discapacitado serán válidas las reglas normales.

4. Se admitirá un reloj de ajedrez especialmente construido para discapacitados visuales. Incorporará las siguientes características:

a). una esfera adaptada con manecillas reforzadas, con un punto resaltado cada cinco minutos y con dos cada quince;

b). una bandera que pueda ser tocada fácilmente. Debería tenerse cuidado para que la bandera esté emplazada de tal manera que permita al jugador tocar la manecilla de los minutos durante los últimos cinco minutos de cada hora.

5. El jugador discapacitado visual debe anotar la partida en Braille o escritura normal, o bien grabar las jugadas mediante un magnetófono.

6. Un lapsus linguae en el anuncio de una jugada debe ser corregido inmediatamente, antes de poner en marcha el reloj del adversario.

7. Si en el transcurso de una partida aparecieran posiciones diferentes en los dos tableros, deben ser corregidas con la ayuda de un árbitro y consultando ambas planillas. Si las dos planillas coinciden, el jugador que ha anotado bien la jugada pero la ha ejecutado mal debe ajustar su posición a la que corresponde con lo escrito.

8. En el caso de que sucedan tales diferencias y se observe que las planillas difieren, se retrocederán las jugadas hasta el punto en que ambas estén de acuerdo y, consecuentemente, el árbitro ajustará los relojes.

9. El jugador discapacitado visual tendrá derecho a valerse de un asistente, quien se encargará de alguna o de todas las funciones siguientes:

a). Realizar la jugada de cualquiera de los jugadores en el tablero del adversario.

b). Anunciar las jugadas de ambos jugadores.

c). Encargarse de anotar la partida por parte del discapacitado visual y poner en marcha el reloj del adversario (teniendo en cuenta la regla 3.c).

d). Informar al discapacitado visual, sólo a requerimiento de éste, del número de jugadas completadas y del tiempo empleado por ambos jugadores.

e). Reclamar la partida en caso de que se haya excedido el límite de tiempo e informar al árbitro cuando el jugador no discapacitado haya tocado una de sus piezas.

f). Llevar a cabo las formalidades necesarias en caso de que se aplace la partida. Si el discapacitado visual no se vale de un asistente, su adversario puede hacer uso de uno que se encargue de las labores mencionadas en los puntos 9.a y b.

5.1.2 Tácticas y Estrategias ontológicas

Las descripciones de las tácticas y estrategias contempladas en nuestra ontología son las siguientes:

Término ontológico	Comentarios
OpeningError	Error in chess opening / Error en la apertura Se trata de una posición donde el oponente ha realizado un movimiento erróneo pero no obvio, sin que el rival haya preparado ningún tipo de trampa (celada). No se trata de una posición realmente táctica o estratégica, pero se incluyen estas posiciones porque se dan con cierta frecuencia en partidas de nivel medio y es interesante conocerlas.
Trap	Chess trap / Celada En esta posición sí se ha preparada una posición táctica denominada “celada”. Si el rival acepta, normalmente, la captura de una pieza, verá en pocos movimientos que ha cometido un error.
Drowned	Drowned / Ahogado En ajedrez, el ahogado es una situación que se produce cuando el jugador de quien es el turno no tiene jugadas legales para realizar y el rey no se encuentra en estado de jaque. Es decir, el rey no puede moverse debido ya sea porque si mueve a esas casillas queda en posición de jaque o por estar ocupadas por piezas propias, o por piezas ajenas que están defendidas, y además no hay otras piezas o peones que puedan moverse o comer a piezas adversarias. A esto también se le conoce como tablas por rey ahogado. La posición en sí no es una táctica. Lo interesante es poder conseguir dicha posición cuando tenemos una partida en clara desventaja (unas tablas son mucho mejor que una derrota).
ContinuousCheck	Continuous Check / Jaque continuo El “jaque continuo” es una de las formas posibles de que una partida de ajedrez termine en tablas. Se da cuando uno de los jugadores está en situación de dar jaque perpetuamente al rey adversario. En este caso, el jugador que da jaque puede reclamar el empate. Actualmente, esta regla no es oficial (no está en el reglamento de la federación) aunque sus efectos quedan incluidos en el final por tercera repetición de la posición y por la regla de los 50 movimientos.
Combinations	Combinations / Combinación En ajedrez, una combinación es una secuencia relativamente larga de movimientos, a menudo iniciada con un sacrificio, que deja al oponente pocas opciones y un resultado prácticamente ganador. En muchos momentos de una partida de ajedrez, cada jugador tiene varias opciones razonables para elegir, lo que hace difícil seguir un plan excepto en términos estratégicos. Las combinaciones, en contra de la norma, son movimientos suficientemente forzados como para que se pueda calcular exactamente cuanta ventaja se conseguirá contra cualquier defensa. Normalmente es necesario analizar varios movimientos detalladamente antes de lanzar una combinación o, si no, el sacrificio inicial no se recuperaría, perdiendo muchas opciones de victoria.
DoubleAttack	The Double Attack or Fork / El Ataque Doble Un ataque doble es el que se produce cuando se amenazan dos piezas del contrario a la vez, de forma que al atacante, en muchos casos, puede capturar una de las dos piezas. El caballo es la mejor pieza para un ataque doble, pero no la única: doble con peón, doble con alfil, doble con

	torre, doble con dama y doble con rey son el resto de opciones.
DiscoveredAttack	Discovered attack / Ataque a la descubierta Consiste en amenazar dos piezas del adversario en una misma jugada con dos piezas nuestras. Amenazamos una con la pieza que hemos movido y otra con una segunda pieza que con el movimiento ha quedado descubierta. Este tipo de jugada táctica se puede hacer con muchísimas combinaciones de piezas. Una variante de este doble consiste en tomar una pieza adversaria, pese a que la tenga defendida. Si nuestro rival nos toma la pieza que hemos movido, entonces podemos tomar otra pieza que nos ha quedado descubierta, y si el rival opta por retirar o defender la pieza amenazada por nuestra pieza descubierta, entonces podremos retirar y salvar la pieza que hemos utilizado para tomar inicialmente. El resultado, en muchos casos, es una ganancia de material.
DiscoveredCheck	Discovered check / Jaque a la descubierta En ajedrez, como hemos visto, un ataque a la descubierta es un ataque producido cuando una pieza se aparta del camino de otra. El jaque a la descubierta es un caso especial de este ataque, donde el ataque desenmascarado es un jaque (si la pieza que se mueve también da un jaque se produce un jaque doble, como veremos a continuación). Los ataques a la descubierta en general y los jaques dobles en particular, pueden ser extremadamente potentes, ya que la pieza movida puede realizar amenazas independientemente de la pieza que cubre. Como muchas tácticas de ajedrez, tienen éxito porque el oponente no puede protegerse de las dos amenazas a la vez.
DoubleCheck	Double check / Jaque doble o Jaque a la descubierta doble Como ya hemos indicado, se produce cuando dos piezas dan jaque a la vez al rey contrario. Para que esto pueda ser posible, una de las piezas ha debido descubrir el ataque de la segunda pieza.
Pinning	Pinning / Clavada En ajedrez, un clavado o clavada es una situación en la cual una pieza es forzada a permanecer en su sitio debido a que su movimiento podría exponer una pieza de más valor a ser capturada. Se denomina “clavada absoluta” cuando la segunda pieza atacada es el rey, dado que bajo estas condiciones no es posible realizar el movimiento de la primera pieza porque dejaría en jaque a su rey (jugada ilegal). Se denomina “clavada relativa” cuando la segunda pieza no es el rey, pero es una pieza de mayor valor que la primera. En esta situación, la primera pieza puede ser movida pero se perdería (en muchas ocasiones, si no tenemos posibilidad de contra-ataque) la segunda.
Skewer	Skewer / Enfilada En ajedrez, una enfilada se produce cuando una pieza ataca dos piezas enemigas situadas en la misma fila, columna o diagonal. Cuando la pieza delantera más valiosa se aparta, la de menos valor puede ser capturada. Las piezas más eficaces para las enfiladas son los alfiles, que crean enfiladas sobre damas y torres, porque no importa que el objetivo final esté defendido, el cambio entre alfil y torre siempre será ventajoso (se suele decir ventaja de calidad). Para que una dama, por ejemplo, lleve a cabo una enfilada con éxito, la pieza del objetivo debe estar, por lo general, indefensa.
Attraction	Attraction / Atracción La atracción es un elemento táctico que consiste en atraer favorablemente una pieza rival (por lo común, el rey) a una casilla en particular. La atracción tiene una gran semejanza con el tema de la desviación que veremos a continuación.

Deflection	Deflection or Deviation / Desviación La desviación es un arma táctica importante; consiste en forzar a una pieza rival a dejar su posición actual, dando a nuestras piezas el acceso a casillas o líneas importantes.
EliminateDefense	Eliminate Defense / Eliminación de la defensa Un arma táctica de gran poder es la combinación basada en la eliminación directa de la pieza que controla líneas o casillas importantes en el tablero. Tiene semejanza con la desviación, pero en este caso la pieza contraria es capturada y no desviada.
Freeing	Freeing / Liberación de líneas o casillas La liberación de líneas (o casillas) es una figura táctica que consiste en despejar determinada(s) casilla(s) o líneas clave mediante sacrificios, para dinamizar a un equipo de piezas que podrán, así, ejecutar una combinación. Las líneas pueden ser diagonales, filas o columnas. La liberación de una casilla puede serlo con fines defensivos u ofensivos. Esa “casilla-clave”, en una maniobra defensiva es, generalmente, una casilla de escape del rey. En el caso ofensivo es una casilla que permitirá instalarse en ella a una de las piezas que tendrá un rol protagonista en la posible combinación. En general, las líneas se abren para luego ser utilizadas.
Interception	Interception / Intercepción de líneas Mediante este recurso táctico, la conexión entre dos o más piezas del oponente es obstruida, impidiendo el acceso a una casilla o línea vital.
Lock	Lock / Bloqueo táctico El objetivo de este recurso táctico consiste en forzar a una pieza rival a ocupar una casilla que resulta perjudicial, por lo común, para su propio rey. Es una forma eficaz de eliminar la coordinación de las fuerzas contrarias.
Xray	X-ray / Rayos X En ajedrez, los rayos X es una táctica de ataque que consiste en que la acción de una pieza está siendo tapada por una pieza del bando contrario, a su vez esta puede ser utilizada para obtener ventaja en el juego en futuras jugadas.
Overload	Overload / Sobrecarga En ajedrez, una pieza que tiene muchas obligaciones defensivas se dice que está sobrecargada. Una pieza sobrecargada a veces puede ser desviada o ser obligada a abandonar una de sus tareas defensivas con el objetivo de provocar una situación de ganancia de material o mate.
WeakLine	Weak Line / Debilidad en línea En ajedrez, se conoce como debilidad de una línea, normalmente la última o penúltima del oponente, a la situación táctica que permite ocupar dicha línea con una pieza provocando situaciones de difícil defensa.
Zwischenzug	Zwischenzug / Jugada Intermedia El zwischenzug (palabra del alemán para "jugada intermedia") es una táctica de ajedrez en que un jugador, en vez de jugar el movimiento esperado (comúnmente una recaptura de una pieza que el

	oponente acaba de capturar) interpola antes otro movimiento, realizando una amenaza intermedia que el oponente tiene que responder y entonces juega el movimiento esperado. Idealmente, el zwischenzug le da ventaja al jugador que lo realiza. Tal movimiento es llamado una intermedia o una jugada intermedia. NOTA: *En algunas ocasiones, en el mundo del ajedrez se utilizan términos en alemán para todos los idiomas. Este es un caso, pero hay muchos otros, como por ejemplo, el término "Zugzwang" (quién mueve pierde).*
Promotion	Promotion or Pawn Promotion / Coronación El peón tiene la virtud de convertirse en cualquier pieza de su mismo color una vez que ha llegado a la octava fila, esto se denomina coronación, y representa un tema importante a considerar en la táctica y estrategia de muchas partidas. Se puede coronar a cualquier pieza, salvo el rey u otro peón, así es posible que llegue a haber hasta nueve damas, varios alfiles, caballos o torres que circulan por casillas del mismo color. La acción de la pieza coronada es inmediata, es decir, si el rey está en la línea de coronación y al coronar un peón se pide una dama o una torre el rey queda automáticamente en jaque, y hasta en jaque mate si no puede eludir la amenaza. En la mayoría de los casos los jugadores prefieren promover el peón a una dama porque esto representa una ventaja decisiva.
AttackKingCenter	Attack on the king in the center / Ataque al rey en el centro Es bien sabido que conviene poner el rey en seguridad lo antes posible. Pero en la partida de ajedrez ya desde el comienzo se plantea una lucha para conseguir diversos objetivos, y la principal dificultad es valorar cual merece prioridad. En la apertura se busca el rápido desarrollo, el dominio del centro, la obtención de buenas casillas para las piezas, sin olvidar la necesaria atención al equilibrio material. Por ello, a veces, la seguridad del rey pasa a un segundo plano, en la creencia de que, mientras su posición sea segura, se puede retrasar el enroque unas jugadas a fin de atender a otros objetivos. Esta estrategia es muchas veces correcta, pero el problema ocurre cuando se sobrepasan los límites en virtud de una falsa valoración de la posición y el rey permanece en el centro más tiempo del debido. Esta táctica busca el ataque al rey cuando permanece sin enroque.
AttackCastledKing	Attack on the castled king / Ataque al rey enrocado A través de la práctica se han ido conociendo diversos sistemas de iniciar y rematar un ataque sobre el rey enrocado. Basándose en posiciones similares, han surgido combinaciones que han allanado el camino al deseado triunfo. Estos esquemas han proporcionado una idea general de cómo puede ejecutarse un asalto al rey, y son estas posiciones y sacrificios los que deben ser catalogados dentro de este elemento.
PawnEndgame	Pawn endgame / Final de peones Dentro del campo de la estrategia y táctica, hay posiciones finales que pueden ser resueltas de manera satisfactoria si conocemos el movimiento exacto a realizar. En esta categoría se incluyen sólo las posiciones donde sólo intervienen peones (uno o varios).
RookEndgame	Rook endgame / Final de torre El mismo comentario que en el final de peones, pero con torres en el tablero.
MinorPieceEndgame	Minor pieces endgame / Final de piezas menores

	El mismo comentario que en el final de peones, pero con piezas menores (alfiles y/o caballos) en el tablero.
QueenEndgame	Queen endgame / Final de dama El mismo comentario que en el final de peones, pero con la dama en el tablero.
OtherEndgame	Other endgame / Otros finales El mismo comentario que en el final de peones, pero con cualquier otra combinación de piezas en el tablero, no incluidas en los casos anteriores.
MateInOne	Mate in one / Mate en un movimiento Posición con posibilidad de dar jaque mate en un solo movimiento.
MateInTwo	Mate in two / Mate en dos movimientos Posición con posibilidad de dar jaque mate en dos movimientos.
MateInThree	Mate in three / Mate en tres movimientos Posición con posibilidad de dar jaque mate en tres movimientos.
OtherMate	Other Mate / Otros mates Posición con posibilidad de dar jaque mate con un número indeterminado de movimientos.

Tabla LXIX: Tácticas y estrategias ontológicas.

A modo de resumen, y tal como ya hicimos en la sección donde se describían las clases de nuestra ontología, se incluye una figura que contiene todas estas tácticas y estrategias:

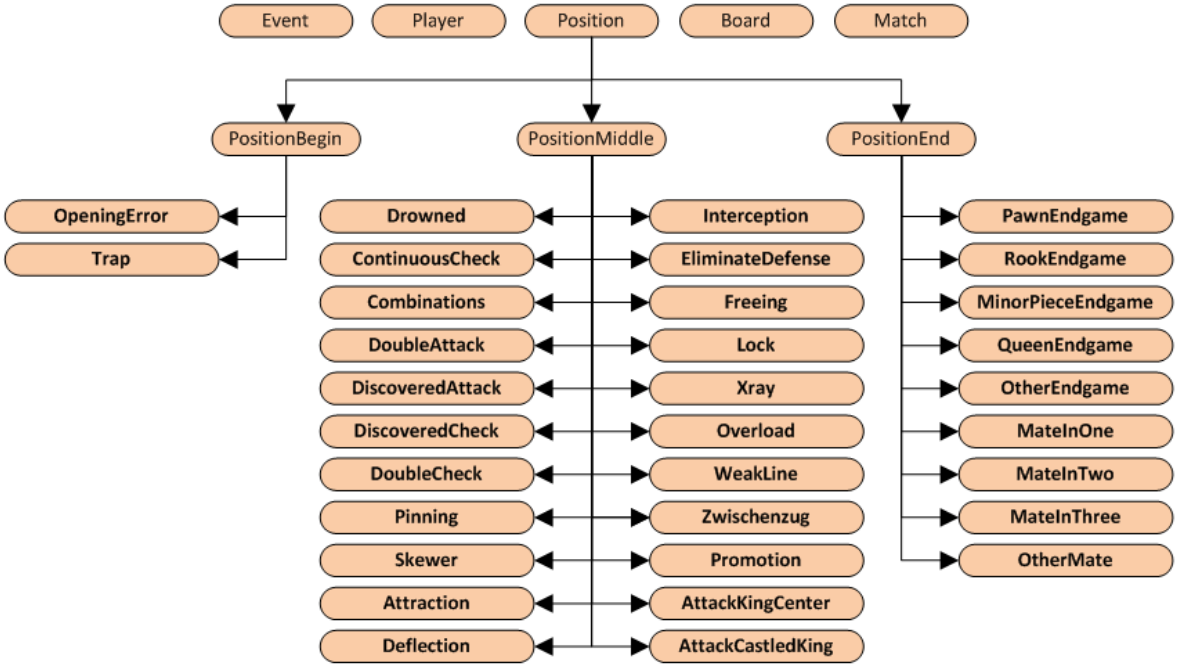


Figura XXXIX: Jerarquía de clases de la ontología ChessPosition.

5.1.3 Notación NAG (Numeric Annotation Glyphs)

Como ya se ha indicado en este documento, los NAGs permiten incluir anotaciones en una partida para incluir comentarios sobre la misma. Los ficheros PGN contienen unos NAGs normalizados que presentaremos en este anexo (presentamos los textos en ingles, dado que se suelen utilizar en este idioma):

NAG	Interpretation
0	null annotation
1	good move (traditional "!")
2	poor move (traditional "?")
3	very good move (traditional "!!")
4	very poor move (traditional "??")
5	speculative move (traditional "!?")
6	questionable move (traditional "?!")
7	forced move (all others lose quickly)
8	singular move (no reasonable alternatives)
9	worst move
10	drawish position
11	equal chances, quiet position
12	equal chances, active position
13	unclear position
14	White has a slight advantage
15	Black has a slight advantage
16	White has a moderate advantage
17	Black has a moderate advantage
18	White has a decisive advantage
19	Black has a decisive advantage
20	White has a crushing advantage (Black should resign)
21	Black has a crushing advantage (White should resign)
22	White is in zugzwang
23	Black is in zugzwang
24	White has a slight space advantage
25	Black has a slight space advantage
26	White has a moderate space advantage
27	Black has a moderate space advantage
28	White has a decisive space advantage
29	Black has a decisive space advantage
30	White has a slight time (development) advantage
31	Black has a slight time (development) advantage
32	White has a moderate time (development) advantage
33	Black has a moderate time (development) advantage
34	White has a decisive time (development) advantage
35	Black has a decisive time (development) advantage

36	White has the initiative
37	Black has the initiative
38	White has a lasting initiative
39	Black has a lasting initiative
40	White has the attack
41	Black has the attack
42	White has insufficient compensation for material deficit
43	Black has insufficient compensation for material deficit
44	White has sufficient compensation for material deficit
45	Black has sufficient compensation for material deficit
46	White has more than adequate compensation for material deficit
47	Black has more than adequate compensation for material deficit
48	White has a slight center control advantage
49	Black has a slight center control advantage
50	White has a moderate center control advantage
51	Black has a moderate center control advantage
52	White has a decisive center control advantage
53	Black has a decisive center control advantage
54	White has a slight kingside control advantage
55	Black has a slight kingside control advantage
56	White has a moderate kingside control advantage
57	Black has a moderate kingside control advantage
58	White has a decisive kingside control advantage
59	Black has a decisive kingside control advantage
60	White has a slight queenside control advantage
61	Black has a slight queenside control advantage
62	White has a moderate queenside control advantage
63	Black has a moderate queenside control advantage
64	White has a decisive queenside control advantage
65	Black has a decisive queenside control advantage
66	White has a vulnerable first rank
67	Black has a vulnerable first rank
68	White has a well protected first rank
69	Black has a well protected first rank
70	White has a poorly protected king
71	Black has a poorly protected king
72	White has a well protected king
73	Black has a well protected king
74	White has a poorly placed king
75	Black has a poorly placed king
76	White has a well placed king
77	Black has a well placed king
78	White has a very weak pawn structure

79	Black has a very weak pawn structure
80	White has a moderately weak pawn structure
81	Black has a moderately weak pawn structure
82	White has a moderately strong pawn structure
83	Black has a moderately strong pawn structure
84	White has a very strong pawn structure
85	Black has a very strong pawn structure
86	White has poor knight placement
87	Black has poor knight placement
88	White has good knight placement
89	Black has good knight placement
90	White has poor bishop placement
91	Black has poor bishop placement
92	White has good bishop placement
93	Black has good bishop placement
84	White has poor rook placement
85	Black has poor rook placement
86	White has good rook placement
87	Black has good rook placement
98	White has poor queen placement
99	Black has poor queen placement
100	White has good queen placement
101	Black has good queen placement
102	White has poor piece coordination
103	Black has poor piece coordination
104	White has good piece coordination
105	Black has good piece coordination
106	White has played the opening very poorly
107	Black has played the opening very poorly
108	White has played the opening poorly
109	Black has played the opening poorly
110	White has played the opening well
111	Black has played the opening well
112	White has played the opening very well
113	Black has played the opening very well
114	White has played the middlegame very poorly
115	Black has played the middlegame very poorly
116	White has played the middlegame poorly
117	Black has played the middlegame poorly
118	White has played the middlegame well
119	Black has played the middlegame well
120	White has played the middlegame very well
121	Black has played the middlegame very well

122	White has played the ending very poorly
123	Black has played the ending very poorly
124	White has played the ending poorly
125	Black has played the ending poorly
126	White has played the ending well
127	Black has played the ending well
128	White has played the ending very well
129	Black has played the ending very well
130	White has slight counterplay
131	Black has slight counterplay
132	White has moderate counterplay
133	Black has moderate counterplay
134	White has decisive counterplay
135	Black has decisive counterplay
136	White has moderate time control pressure
137	Black has moderate time control pressure
138	White has severe time control pressure
139	Black has severe time control pressure

Tabla LXX: Claves notación NAG.

5.1.4 Siglas de países del Comité Olímpico Internacional (COI)

A continuación se incluyen todas las siglas del Comité Olímpico Internacional (COI):

Sigla	País	Sigla	País	Sigla	País	Sigla	País
AFG	Afghanistan	**EST**	Estonia	**MAU**	Mauritania	**SIP**	Singapore
AIR	Aboard aircraft	**FAI**	Faroe Islands	**MEX**	Mexico	**SLV**	Slovenia
ALB	Albania	**FIJ**	Fiji	**MLI**	Mali	**SMA**	San Marino
ALG	Algeria	**FIN**	Finland	**MLT**	Malta	**SPC**	Aboard spacecraft
AND	Andorra	**FRA**	France	**MNC**	Monaco	**SRI**	Sri Lanka
ANG	Angola	**GAM**	Gambia	**MOL**	Moldova	**SUD**	Sudan
ANT	Antigua	**GCI**	Guernsey-Jersey	**MON**	Mongolia	**SUR**	Surinam
ARG	Argentina	**GEO**	Georgia	**MOZ**	Mozambique	**SVE**	Sweden
ARM	Armenia	**GER**	Germany	**MRC**	Morocco	**SWZ**	Switzerland
ATA	Antarctica	**GHA**	Ghana	**MRT**	Mauritius	**SYR**	Syria
AUS	Australia	**GRC**	Greece	**MYN**	Myanmar	**TAI**	Thailand
AZB	Azerbaijan	**GUA**	Guatemala	**NCG**	Nicaragua	**TMT**	Turkmenistan
BAN	Bangladesh	**GUY**	Guyana	**NET**	The Internet	**TRK**	Turkey
BAR	Bahrain	**HAI**	Haiti	**NIG**	Nigeria	**TTO**	Trinidad and Tobago
BHM	Bahamas	**HKG**	Hong Kong	**NLA**	Netherlands Antilles	**TUN**	Tunisia
BEL	Belgium	**HON**	Honduras	**NLD**	Netherlands	**UAE**	United Arab Emirates
BER	Bermuda	**HUN**	Hungary	**NOR**	Norway	**UGA**	Uganda
BIH	Bosnia and Herzegovina	**IND**	India	**NZD**	New Zealand	**UKR**	Ukraine
BLA	Belarus	**IRL**	Ireland	**OST**	Austria	**UNK**	Unknown
BLG	Bulgaria	**IRN**	Iran	**PAK**	Pakistan	**URU**	Uruguay
BLZ	Belize	**IRQ**	Iraq	**PAL**	Palestine	**USA**	United States of America
BOL	Bolivia	**ISD**	Iceland	**PAN**	Panama	**UZB**	Uzbekistan
BRB	Barbados	**ISR**	Israel	**PAR**	Paraguay	**VEN**	Venezuela
BRS	Brazil	**ITA**	Italy	**PER**	Peru	**VGB**	British Virgin Islands
BRU	Brunei	**IVO**	Ivory Coast	**PHI**	Philippines	**VIE**	Vietnam
BSW	Botswana	**JAM**	Jamaica	**PNG**	Papua New Guinea	**VUS**	U.S. Virgin Islands
CAN	Canada	**JAP**	Japan	**POL**	Poland	**WLS**	Wales
CHI	Chile	**JRD**	Jordan	**POR**	Portugal	**YEM**	Yemen
COL	Columbia	**JUG**	Yugoslavia	**PRC**	People's Republic of China	**YUG**	Yugoslavia
CRA	Costa Rica	**KAZ**	Kazakhstan	**PRO**	Puerto Rico	**ZAM**	Zambia
CRO	Croatia	**KEN**	Kenya	**QTR**	Qatar	**ZIM**	Zimbabwe
CSR	Czechoslovakia	**KIR**	Kyrgyzstan	**RIN**	Indonesia	**ZRE**	Zaire
CUB	Cuba	**KUW**	Kuwait	**ROM**	Romania		
CYP	Cyprus	**LAT**	Latvia	**RUS**	Russia		
DEN	Denmark	**LEB**	Lebanon	**SAF**	South Africa		
DOM	Dominican Republic	**LIB**	Libya	**SAL**	El Salvador		
ECU	Ecuador	**LIC**	Liechtenstein	**SCO**	Scotland		
EGY	Egypt	**LTU**	Lithuania	**SEA**	At Sea		
ENG	England	**LUX**	Luxembourg	**SEN**	Senegal		
ESP	Spain	**MAL**	Malaysia	**SEY**	Seychelles		

Tabla LXXI: Claves de países del COI.

5.2 Código fuente

En este apartado se incluyen también a modo de anexos todos los ficheros fuente de los elementos que han intervenido en el desarrollo del generador **GPAHS**. A saber:

Contenido	Sección	Comentarios
Ontología ChessPosition	5.2.1	Fichero OWL con la definición de clases y propiedades realizada en Protégé.
Fichero JLex	5.2.2	Fichero JLex con el analizador léxico y clases principales.
Fichero CUP	5.2.3	Fichero CUP con el analizador sintáctico y resto de elementos del generador.
Fichero PGN	5.2.4	Fichero PGN de ejemplo.
Plantilla HTML	5.2.5	Plantilla HTML de contenido estático.
Ficheros de procesamiento por lotes	5.2.6	Ficheros de procesamiento por lotes.
Hoja de estilos CSS	5.2.7	Hoja de estilos CSS utilizada por la página HTML generada.
Fichero JavaScript	5.2.8	Fichero JavaScript con las funciones JQuery que permiten crear dinamismo en la página.

Tabla LXXII: Secciones del apartado 5.2.

5.2.1 Ontología ChessPosition

En esta primera sección incluimos el contenido completo del fichero “ChessPosition.owl”, el cual contiene la definición OWL formal de nuestra ontología.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.semanticweb.org/ontologies/2011/3/ChessPosition.owl">
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Declaration>
      <Class IRI="#AttackCastledKing"/>
```

```
</Declaration>
<Declaration>
  <Class IRI="#AttackKingCenter"/>
</Declaration>
<Declaration>
  <Class IRI="#Attraction"/>
</Declaration>
<Declaration>
  <Class IRI="#Board"/>
</Declaration>
<Declaration>
  <Class IRI="#Combination"/>
</Declaration>
<Declaration>
  <Class IRI="#ContinuousCheck"/>
</Declaration>
<Declaration>
  <Class IRI="#Deflection"/>
</Declaration>
<Declaration>
  <Class IRI="#DiscoveredAttack"/>
</Declaration>
<Declaration>
  <Class IRI="#DiscoveredCheck"/>
</Declaration>
<Declaration>
  <Class IRI="#DoubleAttack"/>
</Declaration>
<Declaration>
  <Class IRI="#DoubleCheck"/>
</Declaration>
<Declaration>
  <Class IRI="#Drowned"/>
</Declaration>
<Declaration>
  <Class IRI="#EliminateDefense"/>
</Declaration>
<Declaration>
  <Class IRI="#Event"/>
</Declaration>
<Declaration>
  <Class IRI="#Freeing"/>
</Declaration>
<Declaration>
  <Class IRI="#Interception"/>
</Declaration>
<Declaration>
  <Class IRI="#Lock"/>
</Declaration>
<Declaration>
  <Class IRI="#Match"/>
</Declaration>
<Declaration>
  <Class IRI="#MateInOne"/>
</Declaration>
<Declaration>
  <Class IRI="#MateInThree"/>
</Declaration>
<Declaration>
  <Class IRI="#MateInTwo"/>
</Declaration>
<Declaration>
  <Class IRI="#MinorPieceEndgame"/>
</Declaration>
<Declaration>
  <Class IRI="#OpeningError"/>
</Declaration>
<Declaration>
  <Class IRI="#OtherEndgame"/>
</Declaration>
```

```
<Declaration>
  <Class IRI="#OtherMate"/>
</Declaration>
<Declaration>
  <Class IRI="#Overload"/>
</Declaration>
<Declaration>
  <Class IRI="#PawnEndgame"/>
</Declaration>
<Declaration>
  <Class IRI="#Pinning"/>
</Declaration>
<Declaration>
  <Class IRI="#Player"/>
</Declaration>
<Declaration>
  <Class IRI="#Position"/>
</Declaration>
<Declaration>
  <Class IRI="#PositionBegin"/>
</Declaration>
<Declaration>
  <Class IRI="#PositionEnd"/>
</Declaration>
<Declaration>
  <Class IRI="#PositionMiddle"/>
</Declaration>
<Declaration>
  <Class IRI="#Promotion"/>
</Declaration>
<Declaration>
  <Class IRI="#QueenEndgame"/>
</Declaration>
<Declaration>
  <Class IRI="#RookEndgame"/>
</Declaration>
<Declaration>
  <Class IRI="#Skewer"/>
</Declaration>
<Declaration>
  <Class IRI="#Trap"/>
</Declaration>
<Declaration>
  <Class IRI="#WeakLine"/>
</Declaration>
<Declaration>
  <Class IRI="#Xray"/>
</Declaration>
<Declaration>
  <Class IRI="#Zwischenzug"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasBoard"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasEvent"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasMatch"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPlayerBlack"/>
</Declaration>
<Declaration>
  <ObjectProperty IRI="#hasPlayerWhite"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#annotator"/>
</Declaration>
<Declaration>
```

```
    <DataProperty IRI="#boardNum"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#date"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eco"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#elo"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventCountry"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventDate"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventName"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventRounds"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventSponsor"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#eventType"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#fen"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#mode"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#move"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#na"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#name"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#nic"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#opening"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#plyCount"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#result"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#round"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#section"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#setup"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#site"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#stage"/>
```

```
</Declaration>
<Declaration>
  <DataProperty IRI="#subVariation"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#termination"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#time"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#timeControl"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#title"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#typePlayer"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#typePosition"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#uscf"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#utcDate"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#utcTime"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#variation"/>
</Declaration>
<SubClassOf>
  <Class IRI="#AttackCastledKing"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#AttackKingCenter"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Attraction"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Board"/>
  <Class abbreviatedIRI=":Thing"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Combination"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#ContinuousCheck"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Deflection"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#DiscoveredAttack"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#DiscoveredCheck"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
```

```
<SubClassOf>
  <Class IRI="#DoubleAttack"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#DoubleCheck"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Drowned"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#EliminateDefense"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Event"/>
  <Class abbreviatedIRI=":Thing"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Freeing"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Interception"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Lock"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Match"/>
  <Class abbreviatedIRI=":Thing"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#MateInOne"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#MateInThree"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#MateInTwo"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#MinorPieceEndgame"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#OpeningError"/>
  <Class IRI="#PositionBegin"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#OtherEndgame"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#OtherMate"/>
  <Class IRI="#PositionEnd"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#Overload"/>
  <Class IRI="#PositionMiddle"/>
</SubClassOf>
<SubClassOf>
  <Class IRI="#PawnEndgame"/>
```

```
    <Class IRI="#PositionEnd"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Pinning"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Player"/>
    <Class abbreviatedIRI=":Thing"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#PositionBegin"/>
    <Class IRI="#Position"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#PositionEnd"/>
    <Class IRI="#Position"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#PositionMiddle"/>
    <Class IRI="#Position"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Promotion"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#QueenEndgame"/>
    <Class IRI="#PositionEnd"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#RookEndgame"/>
    <Class IRI="#PositionEnd"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Skewer"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Trap"/>
    <Class IRI="#PositionBegin"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#WeakLine"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Xray"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Zwischenzug"/>
    <Class IRI="#PositionMiddle"/>
  </SubClassOf>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#hasBoard"/>
  </FunctionalObjectProperty>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#hasEvent"/>
  </FunctionalObjectProperty>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#hasMatch"/>
  </FunctionalObjectProperty>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#hasPlayerBlack"/>
  </FunctionalObjectProperty>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#hasPlayerWhite"/>
  </FunctionalObjectProperty>
  <ObjectPropertyDomain>
```

```
    <ObjectProperty IRI="#hasBoard"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasBoard"/>
      <Class IRI="#Position"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#hasEvent"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasBoard"/>
      <Class IRI="#Position"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#hasMatch"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasMatch"/>
      <Class IRI="#Position"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#hasPlayerBlack"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasPlayerBlack"/>
      <Class IRI="#Position"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#hasPlayerWhite"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasPlayerWhite"/>
      <Class IRI="#Position"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasBoard"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasBoard"/>
      <Class IRI="#Board"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasEvent"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasEvent"/>
      <Class IRI="#Event"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasMatch"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasMatch"/>
      <Class IRI="#Match"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasPlayerBlack"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasPlayerBlack"/>
      <Class IRI="#Player"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasPlayerWhite"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasPlayerWhite"/>
      <Class IRI="#Player"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
  <FunctionalDataProperty>
```

```
    <DataProperty IRI="#annotator"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#boardNum"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#date"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eco"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#elo"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventCountry"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventDate"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventName"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventRounds"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventSponsor"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#eventType"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#fen"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#mode"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#move"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#na"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#name"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#nic"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#opening"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#plyCount"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#result"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#round"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#section"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#setup"/>
  </FunctionalDataProperty>
  <FunctionalDataProperty>
    <DataProperty IRI="#site"/>
```

```
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#stage"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#subVariation"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#termination"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#time"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#timeControl"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#title"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#typePlayer"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#typePosition"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#uscf"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#utcDate"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#utcTime"/>
</FunctionalDataProperty>
<FunctionalDataProperty>
  <DataProperty IRI="#variation"/>
</FunctionalDataProperty>
<DataPropertyDomain>
  <DataProperty IRI="#annotator"/>
  <Class IRI="#Match"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#annotator"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#annotator"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#boardNum"/>
  <Class IRI="#Board"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#boardNum"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#boardNum"/>
    <Datatype abbreviatedIRI="xsd:integer"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#date"/>
  <Class IRI="#Match"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#date"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#date"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
```

```
<DataPropertyDomain>
  <DataProperty IRI="#eco"/>
  <Class IRI="#Match"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eco"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eco"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#elo"/>
  <Class IRI="#Player"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#elo"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#elo"/>
    <Datatype abbreviatedIRI="xsd:integer"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventCountry"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventCountry"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventCountry"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventDate"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventDate"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventDate"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventName"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventName"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventName"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventRounds"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventRounds"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventRounds"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventSponsor"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
```

```
<DataPropertyDomain>
  <DataProperty IRI="#eventSponsor"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventSponsor"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventType"/>
  <Class IRI="#Event"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#eventType"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#eventType"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#fen"/>
  <Class IRI="#Position"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#fen"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#fen"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#mode"/>
  <Class IRI="#Board"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#mode"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#mode"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#move"/>
  <Class IRI="#Position"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#move"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#move"/>
    <Datatype abbreviatedIRI="xsd:integer"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#na"/>
  <Class IRI="#Player"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#na"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#na"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataSomeValuesFrom>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#name"/>
  <Class IRI="#Player"/>
</DataPropertyDomain>
<DataPropertyDomain>
  <DataProperty IRI="#name"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#name"/>
```

```
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#nic"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#nic"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#nic"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#opening"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#opening"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#opening"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#plyCount"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#plyCount"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#plyCount"/>
      <Datatype abbreviatedIRI="xsd:integer"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#result"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#result"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#result"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#round"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#round"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#round"/>
      <Datatype abbreviatedIRI="xsd:integer"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#section"/>
    <Class IRI="#Event"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#section"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#section"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
```

```
    <DataProperty IRI="#setup"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#setup"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#setup"/>
      <Datatype abbreviatedIRI="xsd:integer"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#site"/>
    <Class IRI="#Event"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#site"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#site"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#stage"/>
    <Class IRI="#Event"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#stage"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#stage"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#subVariation"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#subVariation"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#subVariation"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#termination"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#termination"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#termination"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#time"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#time"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#time"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#timeControl"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
```

```
    <DataProperty IRI="#timeControl"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#timeControl"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#title"/>
    <Class IRI="#Player"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#title"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#title"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#typePlayer"/>
    <Class IRI="#Player"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#typePlayer"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#typePlayer"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#typePosition"/>
    <Class IRI="#Position"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#typePosition"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#typePosition"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#uscf"/>
    <Class IRI="#Player"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#uscf"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#uscf"/>
      <Datatype abbreviatedIRI="xsd:integer"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#utcDate"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#utcDate"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#utcDate"/>
      <Datatype abbreviatedIRI="xsd:string"/>
    </DataSomeValuesFrom>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#utcTime"/>
    <Class IRI="#Match"/>
  </DataPropertyDomain>
  <DataPropertyDomain>
    <DataProperty IRI="#utcTime"/>
    <DataSomeValuesFrom>
      <DataProperty IRI="#utcTime"/>
      <Datatype abbreviatedIRI="xsd:string"/>
```

```
      </DataSomeValuesFrom>
    </DataPropertyDomain>
    <DataPropertyDomain>
      <DataProperty IRI="#variation"/>
      <Class IRI="#Match"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
      <DataProperty IRI="#variation"/>
      <DataSomeValuesFrom>
        <DataProperty IRI="#variation"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
    </DataPropertyDomain>
</Ontology>



<!-- Generated by the OWL API (version 3.2.2.1789) http://owlapi.sourceforge.net -->
```

5.2.2 Fichero Jlex

En esta sección incluimos el contenido completo del fichero JLex.

```
import java.io.*;
import java.util.*;
import java.text.*;
import java_cup.runtime.Symbol;

class Game {
   String event;
   String site;
   String date;
   String round;
   String white;
   String black;
   String result;
   String whiteTitle;
   String blackTitle;
   String whiteElo;
   String blackElo;
   String whiteUSCF;
   String blackUSCF;
   String whiteNA;
   String blackNA;
   String whiteType;
   String blackType;
   String eventDate;
   String eventType;
   String eventCountry;
   String eventRounds;
   String eventSponsor;
   String section;
   String stage;
   String board;
   String opening;
   String variation;
   String subVariation;
   String eco;
   String nic;
   String time;
   String utcTime;
   String utcDate;
   String timeControl;
   String setUp;
   String fen;
   String termination;
   String annotator;
   String mode;
   String plyCount;
   String emaWhite;
   String emaBlack;
   String move;

   public Game () {
      this.event = null;
      this.site = null;
      this.date = null;
      this.round = null;
      this.white = null;
      this.black = null;
      this.result = null;
      this.whiteTitle = null;
      this.blackTitle = null;
      this.whiteElo = null;
      this.blackElo = null;
      this.whiteUSCF = null;
      this.blackUSCF = null;
```

```
      this.whiteNA = null;
      this.blackNA = null;
      this.whiteType = null;
      this.blackType = null;
      this.eventDate = null;
      this.eventType = null;
      this.eventCountry = null;
      this.eventRounds = null;
      this.eventSponsor = null;
      this.section = null;
      this.stage = null;
      this.board = null;
      this.opening = null;
      this.variation = null;
      this.subVariation = null;
      this.eco = null;
      this.nic = null;
      this.time = null;
      this.utcTime = null;
      this.utcDate = null;
      this.timeControl = null;
      this.setUp = null;
      this.fen = null;
      this.termination = null;
      this.annotator = null;
      this.mode = null;
      this.plyCount = null;
      this.emaWhite = null;
      this.emaBlack = null;
      this.move = null;
    }
}

class TablaGames extends LinkedHashMap {

  public void ShowContenido() {
    Enumeration en;
    System.out.println("");
    Iterator it = this.keySet().iterator();
    while (it.hasNext()) {
      String key = (String) it.next();
      Game gm = (Game) this.get(key);
      System.out.println ("Event .....: " + gm.event);
      System.out.println ("Site .....: " + gm.site);
      System.out.println ("Date .....: " + gm.date);
      System.out.println ("Round .....: " + gm.round);
      System.out.println ("White .....: " + gm.white);
      System.out.println ("Black .....: " + gm.black);
      System.out.println ("Result .....: " + gm.result);
      System.out.println ("WhiteTitle ....: " + gm.whiteTitle);
      System.out.println ("BlackTitle ....: " + gm.blackTitle);
      System.out.println ("WhiteElo .....: " + gm.whiteElo);
      System.out.println ("BlackElo .....: " + gm.blackElo);
      System.out.println ("whiteUSCF .....: " + gm.whiteUSCF);
      System.out.println ("BlackUSCF .....: " + gm.blackUSCF);
      System.out.println ("WhiteNA .....: " + gm.whiteNA);
      System.out.println ("BlackNA .....: " + gm.blackNA);
      System.out.println ("WhiteType .....: " + gm.whiteType);
      System.out.println ("BlackType .....: " + gm.blackType);
      System.out.println ("EventDate .....: " + gm.eventDate);
      System.out.println ("EventType .....: " + gm.eventType);
      System.out.println ("EventCountry ..: " + gm.eventCountry);
      System.out.println ("EventRounds ...: " + gm.eventRounds);
      System.out.println ("EventSponsor ..: " + gm.eventSponsor);
      System.out.println ("Section .....: " + gm.section);
      System.out.println ("Stage .....: " + gm.stage);
      System.out.println ("Board .....: " + gm.board);
      System.out.println ("Opening .....: " + gm.opening);
      System.out.println ("Variation .....: " + gm.variation);
      System.out.println ("SubVariation ..: " + gm.subVariation);
```

```
    System.out.println ("ECO ...........: " + gm.eco);
    System.out.println ("NIC ...........: " + gm.nic);
    System.out.println ("Time ..........: " + gm.time);
    System.out.println ("UTCTime .......: " + gm.utcTime);
    System.out.println ("UTCDate .......: " + gm.utcDate);
    System.out.println ("TimeControl ...: " + gm.timeControl);
    System.out.println ("SetUp .........: " + gm.setUp);
    System.out.println ("FEN ...........: " + gm.fen);
    System.out.println ("Termination ...: " + gm.termination);
    System.out.println ("Annotator .....: " + gm.annotator);
    System.out.println ("Mode ..........: " + gm.mode);
    System.out.println ("PlyCount ......: " + gm.plyCount);
    System.out.println ("EmaWhite ......: " + gm.emaWhite);
    System.out.println ("EmaBlack ......: " + gm.emaBlack);
    System.out.println ("Move ..........: " + gm.move);
    System.out.println ("");
   }
  }
}

%%

cadenastring = "\""[^\n"\""]*"\""
string = [a-zA-Z0-9_\-\.\:\/\!\;\,\=\$\#\{\{\?\'\%]+
move = [rnbqkpRNBQKPabcdefg]"x"?[abcdefgh][12345678]("+"|"#")?
digitos = [0-9]+
signo = "+"|"-"
entero = {signo}?{digitos}
nag = "$"{digitos}

%{
        public int getLine(){
                return yyline;
        }
%}

%state MOVE
%notunix
%full
%line
%char
%class lexGPAHS
%cup


%%

<YYINITIAL>[\t \r\n]*           { /* ignorar blancos */
                }
<YYINITIAL>"["    {
        return new Symbol(sym.TK_CORCHETE_ABRIR);
        }
<YYINITIAL>"]"    {
        return new Symbol(sym.TK_CORCHETE_CERRAR);
        }
<YYINITIAL>"*"    {
        yybegin (YYINITIAL);
        return new Symbol(sym.TK_ASTERISCO);
        }
<YYINITIAL>"1-0"  {
        return new Symbol(sym.TK_WIN);
        }
<YYINITIAL>"0-1"  {
        return new Symbol(sym.TK_LOST);
        }
<YYINITIAL>"1/2-1/2" {
        return new Symbol(sym.TK_DRAW);
        }
<YYINITIAL>{cadenastring}  {
            return new Symbol(sym.TK_CADENASTRING, new String(yytext().replaceAll("\"", "")));
```

```
                }
<YYINITIAL>Event       { return new Symbol(sym.TK_EVENT);       }
<YYINITIAL>Site        { return new Symbol(sym.TK_SITE);        }
<YYINITIAL>Date        { return new Symbol(sym.TK_DATE);        }
<YYINITIAL>Round       { return new Symbol(sym.TK_ROUND);       }
<YYINITIAL>White       { return new Symbol(sym.TK_WHITE);       }
<YYINITIAL>Black       { return new Symbol(sym.TK_BLACK);       }
<YYINITIAL>Result      { return new Symbol(sym.TK_RESULT);      }
<YYINITIAL>WhiteTitle  { return new Symbol(sym.TK_WHITETITLE);  }
<YYINITIAL>BlackTitle  { return new Symbol(sym.TK_BLACKTITLE);  }
<YYINITIAL>WhiteElo    { return new Symbol(sym.TK_WHITEELO);    }
<YYINITIAL>BlackElo    { return new Symbol(sym.TK_BLACKELO);    }
<YYINITIAL>WhiteUSCF   { return new Symbol(sym.TK_WHITEUSCF);   }
<YYINITIAL>BlackUSCF   { return new Symbol(sym.TK_BLACKUSCF);   }
<YYINITIAL>WhiteNA     { return new Symbol(sym.TK_WHITENA);     }
<YYINITIAL>BlackNA     { return new Symbol(sym.TK_BLACKNA);     }
<YYINITIAL>WhiteType   { return new Symbol(sym.TK_WHITETYPE);   }
<YYINITIAL>BlackType   { return new Symbol(sym.TK_BLACKTYPE);   }
<YYINITIAL>EventDate   { return new Symbol(sym.TK_EVENTDATE);   }
<YYINITIAL>EventType   { return new Symbol(sym.TK_EVENTTYPE);   }
<YYINITIAL>EventCountry { return new Symbol(sym.TK_EVENTCOUNTRY); }
<YYINITIAL>EventRounds  { return new Symbol(sym.TK_EVENTROUNDS);  }
<YYINITIAL>EventSponsor { return new Symbol(sym.TK_EVENTSPONSOR); }
<YYINITIAL>Section     { return new Symbol(sym.TK_SECTION);     }
<YYINITIAL>Stage       { return new Symbol(sym.TK_STAGE);       }
<YYINITIAL>Board       { return new Symbol(sym.TK_BOARD);       }
<YYINITIAL>Opening     { return new Symbol(sym.TK_OPENING);     }
<YYINITIAL>Variation   { return new Symbol(sym.TK_VARIATION);   }
<YYINITIAL>SubVariation { return new Symbol(sym.TK_SUBVARIATION); }
<YYINITIAL>ECO         { return new Symbol(sym.TK_ECO);         }
<YYINITIAL>NIC         { return new Symbol(sym.TK_NIC);         }
<YYINITIAL>Time        { return new Symbol(sym.TK_TIME);        }
<YYINITIAL>UTCTime     { return new Symbol(sym.TK_UTCTIME);     }
<YYINITIAL>UTCDate     { return new Symbol(sym.TK_UTCDATE);     }
<YYINITIAL>TimeControl  { return new Symbol(sym.TK_TIMECONTROL);  }
<YYINITIAL>SetUp       { return new Symbol(sym.TK_SETUP);       }
<YYINITIAL>FEN         { return new Symbol(sym.TK_FEN);         }
<YYINITIAL>Termination  { return new Symbol(sym.TK_TERMINATION);  }
<YYINITIAL>Annotator   { return new Symbol(sym.TK_ANNOTATOR);    }
<YYINITIAL>Mode        { return new Symbol(sym.TK_MODE);        }
<YYINITIAL>PlyCount    { return new Symbol(sym.TK_PLYCOUNT);    }
<YYINITIAL>EMAWhite    { return new Symbol(sym.TK_EMAWHITE);    }
<YYINITIAL>EMABlack    { return new Symbol(sym.TK_EMABLACK);    }

<YYINITIAL>{string} { yybegin (MOVE);
            return new Symbol(sym.TK_STRING, new String(yytext())); }
<YYINITIAL>.      {
            int line = yyline+1;
            System.out.println("Error lexico en linea " + line + " - " + "Simbolo desconocido en YYINITIAL: " + yytext());
            }

<MOVE>[\t \r\n]*   {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"["        {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"]"        {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"("        {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>")"        {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"{"        {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"}"        {
```

```
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>"1-0"       {
            yybegin (YYINITIAL);
            return new Symbol(sym.TK_WIN);
            }
<MOVE>"0-1"       {
            yybegin (YYINITIAL);
            return new Symbol(sym.TK_LOST);
            }
<MOVE>"1/2-1/2"    {
            yybegin (YYINITIAL);
            return new Symbol(sym.TK_DRAW);
            }
<MOVE>"*"         {
            yybegin (YYINITIAL);
            return new Symbol(sym.TK_ASTERISCO);
            }
<MOVE>{nag}       {
            return new Symbol(sym.TK_NAG, new String(yytext()));
            }
<MOVE>{entero}     {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>{move}      {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>{string}     {
            return new Symbol(sym.TK_STRING, new String(yytext()));
            }
<MOVE>.          {
            int line = yyline+1;
            System.out.println("Error lexico en linea " + line + " - " + "Simbolo desconocido en MOVE: " + yytext());
            }
```

5.2.3 Fichero CUP

En esta sección incluimos el código fuente completo del fichero CUP.

```
import java.io.*;
import java.text.*;
import java.lang.String;
import java.util.*;

parser code
{:
   public static lexGPAHS scanner;
   static ArrayList<String> errors = new ArrayList<String>();
   static int id = 0;
   static String event;
   static String site;
   static String date;
   static String round;
   static String white;
   static String black;
   static String result;
   static String whiteTitle;
   static String blackTitle;
   static String whiteElo;
   static String blackElo;
   static String whiteUSCF;
   static String blackUSCF;
   static String whiteNA;
   static String blackNA;
   static String whiteType;
   static String blackType;
   static String eventDate;
   static String eventType;
   static String eventCountry;
   static String eventRounds;
   static String eventSponsor;
   static String section;
   static String stage;
   static String board;
   static String opening;
   static String variation;
   static String subVariation;
   static String eco;
   static String nic;
   static String time;
   static String utcTime;
   static String utcDate;
   static String timeControl;
   static String setUp;
   static String fen;
   static String termination;
   static String annotator;
   static String mode;
   static String plyCount;
   static String emaWhite;
   static String emaBlack;
   static String move;
   static TablaGames games = new TablaGames();

/*************************************************************************/
/* Limpiar los atributos de un "Game" temporal                           */
/*************************************************************************/

   public static void LimpiarGame() {
      event = null;
      site = null;
      date = null;
      round = null;
```

```
    white = null;
    black = null;
    result = null;
    whiteTitle = null;
    blackTitle = null;
    whiteElo = null;
    blackElo = null;
    whiteUSCF = null;
    blackUSCF = null;
    whiteNA = null;
    blackNA = null;
    whiteType = null;
    blackType = null;
    eventDate = null;
    eventType = null;
    eventCountry = null;
    eventRounds = null;
    eventSponsor = null;
    section = null;
    stage = null;
    board = null;
    opening = null;
    variation = null;
    subVariation = null;
    eco = null;
    nic = null;
    time = null;
    utcTime = null;
    utcDate = null;
    timeControl = null;
    setUp = null;
    fen = null;
    termination = null;
    annotator = null;
    mode = null;
    plyCount = null;
    emaWhite = null;
    emaBlack = null;
    move = null;
  }

/*****************************************************************************/
/* Crear un nuevo elemento del tipo "Game"                                   */
/*****************************************************************************/

  public static void CrearGame(int id) {

    String numCadena = String.valueOf(id);
    Game gm = new Game();
    gm.event = event;
    gm.site = site;
    gm.date = date;
    gm.round = round;
    gm.white = white;
    gm.black = black;
    gm.result = result;
    gm.whiteTitle = whiteTitle;
    gm.blackTitle = blackTitle;
    gm.whiteElo = whiteElo;
    gm.blackElo = blackElo;
    gm.whiteUSCF = whiteUSCF;
    gm.blackUSCF = blackUSCF;
    gm.whiteNA = whiteNA;
    gm.blackNA = blackNA;
    gm.whiteType = whiteType;
    gm.blackType = blackType;
    gm.eventDate = eventDate;
    gm.eventType = eventType;
    gm.eventCountry = eventCountry;
    gm.eventRounds = eventRounds;
```

```
      gm.eventSponsor = eventSponsor;
      gm.section = section;
      gm.stage = stage;
      gm.board = board;
      gm.opening = opening;
      gm.variation = variation;
      gm.subVariation = subVariation;
      gm.eco = eco;
      gm.nic = nic;
      gm.time = time;
      gm.utcTime = utcTime;
      gm.utcDate = utcDate;
      gm.timeControl = timeControl;
      gm.setUp = setUp;
      gm.fen = fen;
      gm.termination = termination;
      gm.annotator = annotator;
      gm.mode = mode;
      gm.plyCount = plyCount;
      gm.emaWhite = emaWhite;
      gm.emaBlack = emaBlack;
      gm.move = move;
      parser.games.put(numCadena, gm);
    }

/*****************************************************************************/
/* Eliminar caracteres no tratables por Protégé                              */
/*****************************************************************************/

   public static String Convertir(String s) {

      if (s != null) {
         String numCadena = s;
         numCadena = numCadena.replaceAll("&", "and");
         return numCadena;
      } else {
         return null;
      }
   }

/*****************************************************************************/
/* Verificacion semantica de la informacion temporal                         */
/*****************************************************************************/

   public static void Semantica() {

/* event */
      if (event != null) {
         if (event.equals("?")) event = null;
         event = Convertir(event);
      }
/* site */
      if (site != null) {
         if (site.equals("?")) site = null;
         site = Convertir(site);
      }
/* date */
      if (date != null) {
         if (date.equals("?")) date = null;
         else if (date.equals("????.??.??")) date = null;
         else date = date.replaceAll("[/.??]", "");
         date = Convertir(date);
      }
/* round */
      if (round != null) {
         if (round.equals("?")) round = null;
         round = Convertir(round);
      }
/* white */
      if (white != null) {
```

```
        if (white.equals("?")) white = null;
        white = Convertir(white);
      }
/* black */
      if (black != null) {
        if (black.equals("?")) black = null;
        black = Convertir(black);
      }
/* result */
      if (result != null) {
        if (result.equals("?")) result = null;
        result = Convertir(result);
      }
/* whiteTitle */
      if (whiteTitle != null) {
        if (whiteTitle.equals("?")) whiteTitle = null;
        whiteTitle = Convertir(whiteTitle);
      }
/* blackTitle */
      if (blackTitle != null) {
        if (blackTitle.equals("?")) blackTitle = null;
        blackTitle = Convertir(blackTitle);
      }
/* whiteElo */
      if (whiteElo != null) {
        if (whiteElo.equals("?")) whiteElo = null;
        whiteElo = Convertir(whiteElo);
      }
/* blackElo */
      if (blackElo != null) {
        if (blackElo.equals("?")) blackElo = null;
        blackElo = Convertir(blackElo);
      }
/* whiteUSCF */
      if (whiteUSCF != null) {
        if (whiteUSCF.equals("?")) whiteUSCF = null;
        whiteUSCF = Convertir(whiteUSCF);
      }
/* blackUSCF */
      if (blackUSCF != null) {
        if (blackUSCF.equals("?")) blackUSCF = null;
        blackUSCF = Convertir(blackUSCF);
      }
/* whiteNA */
      if (whiteNA != null) {
        if (whiteNA.equals("?")) whiteNA = null;
        whiteNA = Convertir(whiteNA);
      }
/* blackNA */
      if (blackNA != null) {
        if (blackNA.equals("?")) blackNA = null;
        blackNA = Convertir(blackNA);
      }
/* whiteType */
      if (whiteType != null) {
        if (whiteType.equals("?")) whiteType = null;
        whiteType = Convertir(whiteType);
      }
/* blackType */
      if (blackType != null) {
        if (blackType.equals("?")) blackType = null;
        blackType = Convertir(blackType);
      }
/* eventDate */
      if (eventDate != null) {
        if (eventDate.equals("?")) eventDate = null;
        else if (eventDate.equals("????.??.??")) eventDate = null;
        else eventDate = eventDate.replaceAll("[/.??]", " ");
        eventDate = Convertir(eventDate);
      }
```

```
/* eventType */
      if (eventType != null) {
         if (eventType.equals("?")) eventType = null;
         eventType = Convertir(eventType);
      }
/* eventCountry */
      if (eventCountry != null) {
         if (eventCountry.equals("?")) eventCountry = null;
         eventCountry = Convertir(eventCountry);
      }
/* eventRounds */
      if (eventRounds != null) {
         if (eventRounds.equals("?")) eventRounds = null;
         eventRounds = Convertir(eventRounds);
      }
/* eventSponsor */
      if (eventSponsor != null) {
         if (eventSponsor.equals("?")) eventSponsor = null;
         eventSponsor = Convertir(eventSponsor);
      }
/* section */
      if (section != null) {
         if (section.equals("?")) section = null;
         section = Convertir(section);
      }
/* stage */
      if (stage != null) {
         if (stage.equals("?")) stage = null;
         stage = Convertir(stage);
      }
/* board */
      if (board != null) {
         if (board.equals("?")) board = null;
         board = Convertir(board);
      }
/* opening */
      if (opening != null) {
         if (opening.equals("?")) opening = null;
         opening = Convertir(opening);
      }
/* variation */
      if (variation != null) {
         if (variation.equals("?")) variation = null;
         variation = Convertir(variation);
      }
/* subVariation */
      if (subVariation != null) {
         if (subVariation.equals("?")) subVariation = null;
         subVariation = Convertir(subVariation);
      }
/* eco */
      if (eco != null) {
         if (eco.equals("?")) eco = null;
         eco = Convertir(eco);
      }
/* nic */
      if (nic != null) {
         if (nic.equals("?")) nic = null;
         nic = Convertir(nic);
      }
/* time */
      if (time != null) {
         if (time.equals("?")) time = null;
         time = Convertir(time);
      }
/* utcTime */
      if (utcTime != null) {
         if (utcTime.equals("?")) utcTime = null;
         utcTime = Convertir(utcTime);
      }
```

```
/* utcDate */
      if (utcDate != null) {
         if (utcDate.equals("?")) utcDate = null;
         else if (utcDate.equals("????.??.??")) utcDate = null;
         else utcDate = utcDate.replaceAll("[/.??]", " ");
         utcDate = Convertir(utcDate);
      }
/* timeControl */
      if (timeControl != null) {
         if (timeControl.equals("?")) timeControl = null;
         timeControl = Convertir(timeControl);
      }
/* setUp */
/* Para el tipo de paginas HTML a generar, el valor "1" o "0" no es significativo */
      if (setUp != null) {
         if (setUp.equals("?")) setUp = null;
         if (setUp.equals("0") || setUp.equals("1") ) setUp = null;
      }
/* fen */
      if (fen != null) {
         if (fen.equals("?")) fen = null;
      }
      if (fen == null) {
         fen = "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1";
      }
/* termination */
      if (termination != null) {
         if (termination.equals("?")) termination = null;
         termination = Convertir(termination);
      }
/* annotator */
      if (annotator != null) {
         if (annotator.equals("?")) annotator = null;
         annotator = Convertir(annotator);
      }
/* mode */
      if (mode != null) {
         if (mode.equals("?")) mode = null;
         mode = Convertir(mode);
      }
/* plyCount */
      if (plyCount != null) {
         if (plyCount.equals("?")) plyCount = null;
         if (plyCount.equals("1")) plyCount = null;
         plyCount = Convertir(plyCount);
      }
/* emaWhite */
      if (emaWhite != null) {
         if (emaWhite.equals("?")) emaWhite = null;
         emaWhite = Convertir(emaWhite);
      }
/* emaBlack */
      if (emaBlack != null) {
         if (emaBlack.equals("?")) emaBlack = null;
         emaBlack = Convertir(emaBlack);
      }
/* move */
      if (move != null) {
         if (move.equals("?")) move = null;
         move = Convertir(move);
      }
    }

/*****************************************************************************/
/* Añadir en MOVE el texto asociado a un código NAG                          */
/*****************************************************************************/

   public static void IncluirNAG(String nag) {

      String temp = null;
```

```
if     (nag.equals("$1"))   temp = "!";
else if (nag.equals("$2"))   temp = "?";
else if (nag.equals("$3"))   temp = "!!";
else if (nag.equals("$4"))   temp = "??";
else if (nag.equals("$5"))   temp = "!?";
else if (nag.equals("$6"))   temp = "?!";
else if (nag.equals("$7"))   temp = "Forced move (all others lose quickly)";
else if (nag.equals("$8"))   temp = "Singular move (no reasonable alternatives)";
else if (nag.equals("$9"))   temp = "Worst move";
else if (nag.equals("$10"))  temp = "Drawish position";
else if (nag.equals("$11"))  temp = "=";
else if (nag.equals("$12"))  temp = "Equal chances, active position";
else if (nag.equals("$13"))  temp = "Unclear position";
else if (nag.equals("$14"))  temp = "+=";
else if (nag.equals("$15"))  temp = "=+";
else if (nag.equals("$16"))  temp = "+/-";
else if (nag.equals("$17"))  temp = "-/+";
else if (nag.equals("$18"))  temp = "+-";
else if (nag.equals("$19"))  temp = "-+";
else if (nag.equals("$20"))  temp = "+--";
else if (nag.equals("$21"))  temp = "--+";
else if (nag.equals("$22"))  temp = "White is in zugzwang";
else if (nag.equals("$23"))  temp = "Black is in zugzwang";
else if (nag.equals("$24"))  temp = "White has a slight space advantage";
else if (nag.equals("$25"))  temp = "Black has a slight space advantage";
else if (nag.equals("$26"))  temp = "White has a moderate space advantage";
else if (nag.equals("$27"))  temp = "Black has a moderate space advantage";
else if (nag.equals("$28"))  temp = "White has a decisive space advantage";
else if (nag.equals("$29"))  temp = "Black has a decisive space advantage";
else if (nag.equals("$30"))  temp = "White has a slight time (development) advantage";
else if (nag.equals("$31"))  temp = "Black has a slight time (development) advantage";
else if (nag.equals("$32"))  temp = "White has a moderate time (development) advantage";
else if (nag.equals("$33"))  temp = "Black has a moderate time (development) advantage";
else if (nag.equals("$34"))  temp = "White has a decisive time (development) advantage";
else if (nag.equals("$35"))  temp = "Black has a decisive time (development) advantage";
else if (nag.equals("$36"))  temp = "White has the initiative";
else if (nag.equals("$37"))  temp = "Black has the initiative";
else if (nag.equals("$38"))  temp = "White has a lasting initiative";
else if (nag.equals("$39"))  temp = "Black has a lasting initiative";
else if (nag.equals("$40"))  temp = "White has the attack";
else if (nag.equals("$41"))  temp = "Black has the attack";
else if (nag.equals("$42"))  temp = "White has insufficient compensation for material deficit";
else if (nag.equals("$43"))  temp = "Black has insufficient compensation for material deficit";
else if (nag.equals("$44"))  temp = "White has sufficient compensation for material deficit";
else if (nag.equals("$45"))  temp = "Black has sufficient compensation for material deficit";
else if (nag.equals("$46"))  temp = "White has more than adequate compensation for material deficit";
else if (nag.equals("$47"))  temp = "Black has more than adequate compensation for material deficit";
else if (nag.equals("$48"))  temp = "White has a slight center control advantage";
else if (nag.equals("$49"))  temp = "Black has a slight center control advantage";
else if (nag.equals("$50"))  temp = "White has a moderate center control advantage";
else if (nag.equals("$51"))  temp = "Black has a moderate center control advantage";
else if (nag.equals("$52"))  temp = "White has a decisive center control advantage";
else if (nag.equals("$53"))  temp = "Black has a decisive center control advantage";
else if (nag.equals("$54"))  temp = "White has a slight kingside control advantage";
else if (nag.equals("$55"))  temp = "Black has a slight kingside control advantage";
else if (nag.equals("$56"))  temp = "White has a moderate kingside control advantage";
else if (nag.equals("$57"))  temp = "Black has a moderate kingside control advantage";
else if (nag.equals("$58"))  temp = "White has a decisive kingside control advantage";
else if (nag.equals("$59"))  temp = "Black has a decisive kingside control advantage";
else if (nag.equals("$60"))  temp = "White has a slight queenside control advantage";
else if (nag.equals("$61"))  temp = "Black has a slight queenside control advantage";
else if (nag.equals("$62"))  temp = "White has a moderate queenside control advantage";
else if (nag.equals("$63"))  temp = "Black has a moderate queenside control advantage";
else if (nag.equals("$64"))  temp = "White has a decisive queenside control advantage";
else if (nag.equals("$65"))  temp = "Black has a decisive queenside control advantage";
else if (nag.equals("$66"))  temp = "White has a vulnerable first rank";
else if (nag.equals("$67"))  temp = "Black has a vulnerable first rank";
else if (nag.equals("$68"))  temp = "White has a well-protected first rank";
else if (nag.equals("$69"))  temp = "Black has a well-protected first rank";
else if (nag.equals("$70"))  temp = "White has a poorly protected king";
```

```
else if (nag.equals("$71"))  temp = "Black has a poorly protected king";
else if (nag.equals("$72"))  temp = "White has a well-protected king";
else if (nag.equals("$73"))  temp = "Black has a well-protected king";
else if (nag.equals("$74"))  temp = "White has a poorly placed king";
else if (nag.equals("$75"))  temp = "Black has a poorly placed king";
else if (nag.equals("$76"))  temp = "White has a well-placed king";
else if (nag.equals("$77"))  temp = "Black has a well-placed king";
else if (nag.equals("$78"))  temp = "White has a very weak pawn structure";
else if (nag.equals("$79"))  temp = "Black has a very weak pawn structure";
else if (nag.equals("$80"))  temp = "White has a moderately weak pawn structure";
else if (nag.equals("$81"))  temp = "Black has a moderately weak pawn structure";
else if (nag.equals("$82"))  temp = "White has a moderately strong pawn structure";
else if (nag.equals("$83"))  temp = "White has a moderately strong pawn structure";
else if (nag.equals("$84"))  temp = "White has a very strong pawn structure";
else if (nag.equals("$85"))  temp = "Black has a very strong pawn structure";
else if (nag.equals("$86"))  temp = "White has poor knight placement";
else if (nag.equals("$87"))  temp = "Black has poor knight placement";
else if (nag.equals("$88"))  temp = "White has good knight placement";
else if (nag.equals("$89"))  temp = "Black has good knight placement";
else if (nag.equals("$90"))  temp = "White has poor bishop placement";
else if (nag.equals("$91"))  temp = "Black has poor bishop placement";
else if (nag.equals("$92"))  temp = "White has good bishop placement";
else if (nag.equals("$93"))  temp = "Black has good bishop placement";
else if (nag.equals("$94"))  temp = "White has poor rook placement";
else if (nag.equals("$95"))  temp = "Black has poor rook placement";
else if (nag.equals("$96"))  temp = "White has good rook placement";
else if (nag.equals("$97"))  temp = "Black has good rook placement";
else if (nag.equals("$98"))  temp = "White has poor queen placement";
else if (nag.equals("$99"))  temp = "Black has poor queen placement";
else if (nag.equals("$100")) temp = "White has good queen placement";
else if (nag.equals("$101")) temp = "Black has good queen placement";
else if (nag.equals("$102")) temp = "White has poor piece coordination";
else if (nag.equals("$103")) temp = "Black has poor piece coordination";
else if (nag.equals("$104")) temp = "White has good piece coordination";
else if (nag.equals("$105")) temp = "Black has good piece coordination";
else if (nag.equals("$106")) temp = "White has played the opening very poorly";
else if (nag.equals("$107")) temp = "Black has played the opening very poorly";
else if (nag.equals("$108")) temp = "White has played the opening poorly";
else if (nag.equals("$109")) temp = "Black has played the opening poorly";
else if (nag.equals("$110")) temp = "White has played the opening well";
else if (nag.equals("$111")) temp = "Black has played the opening well";
else if (nag.equals("$112")) temp = "White has played the opening very well";
else if (nag.equals("$113")) temp = "Black has played the opening very well";
else if (nag.equals("$114")) temp = "White has played the middlegame very poorly";
else if (nag.equals("$115")) temp = "Black has played the middlegame very poorly";
else if (nag.equals("$116")) temp = "White has played the middlegame poorly";
else if (nag.equals("$117")) temp = "Black has played the middlegame poorly";
else if (nag.equals("$118")) temp = "White has played the middlegame well";
else if (nag.equals("$119")) temp = "Black has played the middlegame well";
else if (nag.equals("$120")) temp = "White has played the middlegame very well";
else if (nag.equals("$121")) temp = "Black has played the middlegame very well";
else if (nag.equals("$122")) temp = "White has played the ending very poorly";
else if (nag.equals("$123")) temp = "Black has played the ending very poorly";
else if (nag.equals("$124")) temp = "White has played the ending poorly";
else if (nag.equals("$125")) temp = "Black has played the ending poorly";
else if (nag.equals("$126")) temp = "White has played the ending well";
else if (nag.equals("$127")) temp = "Black has played the ending well";
else if (nag.equals("$128")) temp = "White has played the ending very well";
else if (nag.equals("$129")) temp = "Black has played the ending very well";
else if (nag.equals("$130")) temp = "White has slight counterplay";
else if (nag.equals("$131")) temp = "Black has slight counterplay";
else if (nag.equals("$132")) temp = "White has moderate counterplay";
else if (nag.equals("$133")) temp = "Black has moderate counterplay";
else if (nag.equals("$134")) temp = "White has decisive counterplayº";
else if (nag.equals("$135")) temp = "Black has decisive counterplay";
else if (nag.equals("$136")) temp = "White has moderate time control pressure";
else if (nag.equals("$137")) temp = "Black has moderate time control pressure";
else if (nag.equals("$138")) temp = "White has severe time control pressure";
else if (nag.equals("$139")) temp = "Black has severe time control pressure";
else temp = nag;
```

```
      if (parser.move == null) {
         parser.move = temp;
      } else {
         parser.move = parser.move.concat(temp);
      }
   }

/*****************************************************************************/
/* metodo principal "main"                                                   */
/*****************************************************************************/

public static void main (String argv[]) throws Exception {
   String sFuente;
   File fFuente;
   String[] fuenteSplit;
   String delimitador = "\\.";
   Boolean bloqueEncontrado = null;

   String sPlantillaHTML;
   File fPlantillaHTML = null;
   FileReader frPlantillaHTML = null;
   BufferedReader brPlantillaHTML = null;

   String sDestinoHTML = null;
   File fDestinoHTML = null;
   FileOutputStream fosDestinoHTML = null;
   OutputStreamWriter oswDestinoHTML = null;
   BufferedWriter bwDestinoHTML = null;

   String sPlantillaOWL;
   File fPlantillaOWL = null;
   FileReader frPlantillaOWL = null;
   BufferedReader brPlantillaOWL = null;

   String sDestinoOWL = null;
   File fDestinoOWL = null;
   FileOutputStream fosDestinoOWL = null;
   OutputStreamWriter oswDestinoOWL = null;
   BufferedWriter bwDestinoOWL = null;

   String sLine;
   int indice = 0;
   int ind = 0;

   String sTraza;
   String sPosicion;
   parser Scaner;


/*****************************************************************************/
/* validación número de argumentos de llamada                                */
/*****************************************************************************/
/* Primer argumento ...: Fichero pgn a transformar.                          */
/* Segundo argumento ..: Plantilla HTML.                                     */
/* Tercer argumento ...: Plantilla OWL (ontología).                          */
/* Cuarto argumento ...: nombre de la tactica o estrategia.                  */
/* Quinto argumento ...: Traza activa ??? (true/false).                      */
/*****************************************************************************/

   if (argv.length !=5) {
      System.out.println ("Error en la invocacion del proceso. Uso:");
      System.out.println ("\tjava parser <arg1> <arg2> <arg3> <arg4> <arg5>");
      System.out.println ("\tPrimer argumento ...: Fichero PGN a transformar, con extension \"pgn\".");
      System.out.println ("\tSegundo argumento ..: Plantilla HTML.");
      System.out.println ("\tTercer argumento ...: Plantilla OWL (ontologia).");
      System.out.println ("\tCuarto argumento ...: Tactica/Estrategia.");
      System.out.println ("\tQuinto argumento ...: Traza (true/false).");
      return;
   }
```

```
/*****************************************************************************/
/* ARGUMENTO 1: FICHERO FUENTE PGN. Validaciones                             */
/* 1.- El fichero debe tener la extension "pgn".                             */
/* 2.- El fichero "pgn" debe existir en la subcarpeta "pgn"                  */
/* 3.- No se debe incuir el path de acceso al fichero                        */
/*****************************************************************************/

/* validación de formato y existencia de fichero */
   sFuente = argv[0];
   if (!sFuente.toUpperCase().endsWith(".PGN")) {
      System.out.println ("La extension .PGN es obligatoria en el fichero fuente");
      System.out.println ("Por favor, renombre el fichero " + sFuente + " y lance de nuevo el proceso");
   return;
   }
   sFuente = ".\\pgn\\";
   sFuente = sFuente.concat(argv[0]);
   fFuente = new File(sFuente);
   if (!fFuente.exists()) {
      System.out.println ("El fichero PGN no existe.");
      System.out.println ("Para poder ejecutar el proceso GPAHS se requiere un fichero PGN.");
      return;
   }

/*****************************************************************************/
/* ARGUMENTO 2: FICHERO PLANTILLA HTML                                       */
/*****************************************************************************/

/* validación formato de fichero */
   sPlantillaHTML = argv[1];
   if (!sPlantillaHTML.toUpperCase().endsWith(".HTML")) {
      System.out.println ("La extension .HTML es obligatoria en el fichero plantilla");
      System.out.println ("Por favor, renombre el fichero " + sPlantillaHTML + " y lance de nuevo el proceso");
      return;
   }

/* Si el fichero plantilla no existe el proceso se cancela. */
      fPlantillaHTML = new File(sPlantillaHTML);
      if (!fPlantillaHTML.exists()) {
         System.out.println ("El fichero plantilla no existe.");
         System.out.println ("Para poder ejecutar el proceso GPAHS se requiere una plantilla HTML.");
         return;
      } else {
         frPlantillaHTML = new FileReader(sPlantillaHTML);
         brPlantillaHTML = new BufferedReader(frPlantillaHTML);
      }

/*****************************************************************************/
/* ARGUMENTO 3: FICHERO PLANTILLA OWL                                        */
/*****************************************************************************/

/* validación formato de fichero */
   sPlantillaOWL = argv[2];
   if (!sPlantillaOWL.toUpperCase().endsWith(".OWL")) {
      System.out.println ("La extension .OWL es obligatoria en el fichero plantilla");
      System.out.println ("Por favor, renombre el fichero " + sPlantillaOWL + " y lance de nuevo el proceso");
      return;
   }

/* Si el fichero plantilla no existe el proceso se cancela. */
      fPlantillaOWL = new File(sPlantillaOWL);
      if (!fPlantillaOWL.exists()) {
         System.out.println ("El fichero plantilla no existe.");
         System.out.println ("Para poder ejecutar el proceso GPAHS se requiere una plantilla OWL.");
         return;
      } else {
         frPlantillaOWL = new FileReader(sPlantillaOWL);
         brPlantillaOWL = new BufferedReader(frPlantillaOWL);
      }

/*****************************************************************************/
```

```
/* ARGUMENTO 3: NOMBRE DE LA TACTICA O ESTRATEGIA                    */
/*************************************************************************/

    sPosicion = argv[3];
    if (!sPosicion.equals("Position")        &&
       !sPosicion.equals("PositionBegin")    &&
       !sPosicion.equals("PositionMiddle")   &&
       !sPosicion.equals("PositionEnd")      &&
       !sPosicion.equals("OpeningError")     &&
       !sPosicion.equals("Trap")             &&
       !sPosicion.equals("Drowned")          &&
       !sPosicion.equals("ContinuousCheck")  &&
       !sPosicion.equals("Combination")      &&
       !sPosicion.equals("DoubleAttack")     &&
       !sPosicion.equals("DiscoveredAttack") &&
       !sPosicion.equals("DiscoveredCheck")  &&
       !sPosicion.equals("DoubleCheck")      &&
       !sPosicion.equals("Pinning")          &&
       !sPosicion.equals("Skewer")           &&
       !sPosicion.equals("Deflection")       &&
       !sPosicion.equals("Interception")     &&
       !sPosicion.equals("EliminateDefense") &&
       !sPosicion.equals("Freeing")          &&
       !sPosicion.equals("Lock")             &&
       !sPosicion.equals("Xray")             &&
       !sPosicion.equals("Overload")         &&
       !sPosicion.equals("WeakLine")         &&
       !sPosicion.equals("Zwischenzug")      &&
       !sPosicion.equals("Promotion")        &&
       !sPosicion.equals("AttackKingCenter") &&
       !sPosicion.equals("AttackCastledKing") &&
       !sPosicion.equals("PawnEndgame")      &&
       !sPosicion.equals("RookEndgame")      &&
       !sPosicion.equals("MinorPieceEndgame") &&
       !sPosicion.equals("QueenEndgame")     &&
       !sPosicion.equals("OtherEndgame")     &&
       !sPosicion.equals("MateInOne")        &&
       !sPosicion.equals("MateInTwo")        &&
       !sPosicion.equals("MateInThree")      &&
       !sPosicion.equals("OtherMate")) {
       System.out.println ("La posicion indicada en el argumento no es valida para la ontologia definida.");
       System.out.println ("Por favor, cambie " + sPosicion + " y lance de nuevo el proceso.");
       return;
    }

/*************************************************************************/
/* ARGUMENTO 4: TRAZA                                                    */
/*************************************************************************/

    sTraza = argv[4];
    if (!sTraza.equals("true") &&
       !sTraza.equals("false")) {
       System.out.println ("El valor de traza debe ser \"true\" o \"false\".");
       System.out.println ("Por favor, cambie " + sTraza + " y lance de nuevo el proceso.");
       return;
    }

/*************************************************************************/
/* SCANER                                                                */
/*************************************************************************/

  try {
     scanner = new lexGPAHS(new FileInputStream(sFuente));
  } catch (Exception ioe) {
     System.out.println("Error de lectura en el fichero " + sFuente);
            return;
  }

/* limpiamos un juego  */
  LimpiarGame();
```

```
/* llamada al analizador léxico */
   Scaner = new parser(scanner);
   Scaner.parse();

/* Mostrar errores o mensaje de OK */
/* En el caso de errores no se continua con la ejecución del proceso */
   if(Scaner.errors.size() > 0) {
      System.out.println("Programa Incorrecto. Numero de errores: " + Scaner.errors.size());
      Iterator it = Scaner.errors.iterator();
      while(it.hasNext()) {
         String error = (String)it.next();
         System.out.println(error);
         return;
      }
   } else {
      System.out.println("Programa OK !!!");
   }

/*****************************************************************************/
/* Creación del fichero HTML destino a partir de la información recopilada   */
/*****************************************************************************/

   try {
      fuenteSplit = argv[0].split(delimitador);
      sDestinoHTML = ".\\html\\";
      sDestinoHTML = sDestinoHTML.concat(fuenteSplit[0]);
      sDestinoHTML = sDestinoHTML.concat(".html");
      fDestinoHTML = new File(sDestinoHTML);

/* Si el fichero destino existe el proceso se cancela. */
      if (fDestinoHTML.exists()) {
         System.out.println ("El fichero destino HTML ya existe.");
         System.out.println ("Elimine o renombre el fichero antes de lanzar de nuevo el proceso.");
         System.out.println ("Nombre del fichero .....: " + fDestinoHTML.getName());
         System.out.println ("Nombre del fichero completo ..: " + fDestinoHTML.getCanonicalPath());
         System.out.println ("Longitud .....: " + fDestinoHTML.length() + " bytes.");
         return;
      } else {
         fosDestinoHTML = new FileOutputStream(sDestinoHTML);
         oswDestinoHTML = new OutputStreamWriter(fosDestinoHTML, "UTF-8");
         bwDestinoHTML = new BufferedWriter(oswDestinoHTML);
      }

/* leemos el fichero plantilla hasta encontrar null */
      while ((sLine = brPlantillaHTML.readLine())!=null) {
         bloqueEncontrado = false; /* reiniciamos la variable */

/*****************************************************************************
Si hemos alcanzado el bloque 1:
Hay que incluir las invocaciones JavaScript para dibujar los tableros.
A continuación se muestra un ejemplo del contenido a generar:

      var ctx1 = document.getElementById("canvas1").getContext('2d');
      posicion(ctx1, "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq e7 0 1");

Comentarios:

- Se crearán tantas variables "ctxn" como tableros tengamos.
- Los elementos "canvasn" también serán consecutivos.
- El segundo argumento de la invocación a "posición" es el contenido de FEN.
*****************************************************************************/
         if (sLine.startsWith("//<GPAHS_INSERT_1>")) {

            bloqueEncontrado = true; /* hemos encontrado el bloque 1 */
            indice = 0;
            Iterator it = games.keySet().iterator();
            while (it.hasNext()) {
               indice += 1;
               String key = (String) it.next();
```

```
            Game gm = (Game) games.get(key);
            bwDestinoHTML.write("");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("// posicion " + indice + ":");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("var ctx" + indice + " = document.getElementById(\"canvas" + indice + "\").getContext('2d');");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("posicion(ctx" + indice + ", \"" + gm.fen + "\");");
            bwDestinoHTML.newLine();
          } /* while */
          ind = indice;
        } /* if bloque 1 */


/*****************************************************************************
Si hemos alcanzado el bloque 2:
Hay que incluir los elementos canvas de la página, así como la información
extra de la partida. A continuación se muestra un ejemplo del contenido a generar:

  <hr />
  <div class="posicion">
    <p class="posicion_nombre">Position 1</p>
    <table border=".5">
    <tr><td>
    <p class="posicion_grafico">
      <canvas id="canvas1" width="390" height="300">
        Your browser does not accept the HTML5 canvas element.
      </canvas>
    </p></td>
    <td valign="top" align="right">Texto de la derecha, en el top</td>
    </table>
    <p class="posicion_solucion">solución ...</p>
    <ul class="actions">
      <li class="solucion"><a href="" title="Solución">Solución</a></li>
    </ul>
  </div>

*****************************************************************************/

        if (sLine.startsWith("<!--<GPAHS_INSERT_2>-->")) {
          bloqueEncontrado = true; /* hemos encontrado el bloque 2 */
          indice = 0;
          Iterator it = games.keySet().iterator();
          while (it.hasNext()) {
            indice += 1;
            String key = (String) it.next();
            Game gm = (Game) games.get(key);
            bwDestinoHTML.write("<hr />");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("<div class=\"posicion\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <p class=\"posicion_nombre\">Position " + indice + "</p>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <table id=\"data1\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <tr><td rowspan=\"99\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <p class=\"posicion_gráfico\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("    <canvas id=\"canvas" + indice + "\" width=\"390\" height=\"300\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("      Your browser does not accept the HTML5 canvas element.");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("    </canvas>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  </p></td>");
            bwDestinoHTML.newLine();
            if (gm.event != null) {
              bwDestinoHTML.write("<tr><td id=\"data2\">Event ..:</td><td>" + gm.event + "</td></tr>");
              bwDestinoHTML.newLine();
```

```
}
if (gm.site != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Site ..:</td><td>" + gm.site + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.date != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Date ..:</td><td>" + gm.date + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.round != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Round ..:</td><td>" + gm.round + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.white != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White ..:</td><td>" + gm.white + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.black != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black ..:</td><td>" + gm.black + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.result != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Result ..:</td><td>" + gm.result + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.whiteTitle != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White Title ..:</td><td>" + gm.whiteTitle + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.blackTitle != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black Title ..:</td><td>" + gm.blackTitle + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.whiteElo != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White ELO ..:</td><td>" + gm.whiteElo + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.blackElo != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black ELO ..:</td><td>" + gm.blackElo + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.whiteUSCF != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White USCF ..:</td><td>" + gm.whiteUSCF + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.blackUSCF != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black USCF ..:</td><td>" + gm.blackUSCF + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.whiteNA != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White NA ..:</td><td>" + gm.whiteNA + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.blackNA != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black NA ..:</td><td>" + gm.blackNA + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.whiteType != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">White Type ..:</td><td>" + gm.whiteType + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.blackType != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Black Type ..:</td><td>" + gm.blackType + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.eventDate != null) {
   bwDestinoHTML.write("<tr><td id=\"data2\">Event Date ..:</td><td>" + gm.eventDate + "</td></tr>");
   bwDestinoHTML.newLine();
}
if (gm.eventType != null) {
```

```
    bwDestinoHTML.write("<tr><td id=\"data2\">Event Type ..:</td><td>" + gm.eventType + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.eventCountry != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Event Country ..:</td><td>" + gm.eventCountry + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.eventRounds != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Event Rounds ..:</td><td>" + gm.eventRounds + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.eventSponsor != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Event Sponsor ..:</td><td>" + gm.eventSponsor + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.section != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Section ..:</td><td>" + gm.section + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.stage != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Stage ..:</td><td>" + gm.stage + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.board != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Board ..:</td><td>" + gm.board + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.opening != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Opening ..:</td><td>" + gm.opening + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.variation != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Variation ..:</td><td>" + gm.variation + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.subVariation != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Sub Variation ..:</td><td>" + gm.subVariation + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.eco != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">ECO ..:</td><td>" + gm.eco + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.nic != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">NIC ..:</td><td>" + gm.nic + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.time != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Time ..:</td><td>" + gm.time + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.utcTime != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">UTC Time ..:</td><td>" + gm.utcTime + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.utcDate != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">UTC Date ..:</td><td>" + gm.utcDate + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.timeControl != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">Time Control ..:</td><td>" + gm.timeControl + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.setUp != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">SetUp ..:</td><td>" + gm.setUp + "</td></tr>");
    bwDestinoHTML.newLine();
  }
  if (gm.fen != null) {
    bwDestinoHTML.write("<tr><td id=\"data2\">FEN ..:</td><td>" + gm.fen + "</td></tr>");
    bwDestinoHTML.newLine();
```

```
            }
            if (gm.termination != null) {
               bwDestinoHTML.write("<tr ><td id=\"data2\">Termination ..:</td><td>" + gm.termination + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            if (gm.annotator != null) {
               bwDestinoHTML.write("<tr><td id=\"data2\">Annotator ..:</td><td>" + gm.annotator + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            if (gm.mode != null) {
               bwDestinoHTML.write("<tr><td id=\"data2\">mode ..:</td><td>" + gm.mode + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            if (gm.plyCount != null) {
               bwDestinoHTML.write("<tr><td id=\"data2\">plyCount ..:</td><td>" + gm.plyCount + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            if (gm.emaWhite != null) {
               bwDestinoHTML.write("<tr><td id=\"data2\">emaWhite ..:</td><td>" + gm.emaWhite + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            if (gm.emaBlack != null) {
               bwDestinoHTML.write("<tr><td id=\"data2\">emaBlack ..:</td><td>" + gm.emaBlack + "</td></tr>");
               bwDestinoHTML.newLine();
            }
            bwDestinoHTML.write("  </table>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("<br />");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <p class=\"posicion_solucion\">" + gm.move + "</p>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  <ul class=\"actions\">");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("    <li class=\"solucion\"><a href=\"\" title=\"Solución\">Solution</a></li>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("  </ul>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("</div>");
            bwDestinoHTML.newLine();
            bwDestinoHTML.write("");
            bwDestinoHTML.newLine();
          } /* while */
        }

/*********************************************************************************
Si no hemos encontrado ningún bloque de sustitución:
Hay que incluir la linea leida.
*********************************************************************************/

        if (!bloqueEncontrado) {
           bwDestinoHTML.write(sLine);
           bwDestinoHTML.newLine();
        }

      } /* while */

      System.out.println("Fichero de entrada PGN .....: " + sFuente);
      System.out.println("Posiciones HTML generadas ..: " + ind + " en el fichero " + sDestinoHTML);

/*********************************************************************************
Control de errores fichero HTML
*********************************************************************************/

    } catch(Exception e){
       e.printStackTrace();
    } finally {
// En el finally cerramos el fichero, para asegurarnos
// que se cierra tanto si todo va bien como si salta
// una excepcion.
       try {
```

```
        if( null != frPlantillaHTML ){
          frPlantillaHTML.close();
        }
        if( null != bwDestinoHTML ){
          bwDestinoHTML.flush();
          bwDestinoHTML.close();
        }
      } catch (Exception e2) {
        e2.printStackTrace();
      } /* catch */
    } /* finally */



/*****************************************************************************/
/* Creación del fichero OWL destino a partir de la información recopilada    */
/*****************************************************************************/

    try {
      fuenteSplit = argv[0].split(delimitador);
      sDestinoOWL = ".\\owl\\";
      sDestinoOWL = sDestinoOWL.concat(fuenteSplit[0]);
      sDestinoOWL = sDestinoOWL.concat(".owl");
      fDestinoOWL = new File(sDestinoOWL);

/* Si el fichero destino existe el proceso se cancela. */
      if (fDestinoOWL.exists()) {
        System.out.println ("El fichero destino OWL ya existe.");
        System.out.println ("Elimine o renombre el fichero antes de lanzar de nuevo el proceso.");
        System.out.println ("Nombre del fichero .....: " + fDestinoOWL.getName());
        System.out.println ("Nombre del fichero completo ..: " + fDestinoOWL.getCanonicalPath());
        System.out.println ("Longitud ..............: " + fDestinoOWL.length() + " bytes.");
        return;
      } else {
        fosDestinoOWL = new FileOutputStream(sDestinoOWL);
        oswDestinoOWL = new OutputStreamWriter(fosDestinoOWL, "UTF-8");
        bwDestinoOWL = new BufferedWriter(oswDestinoOWL);
      }

/*****************************************************************************/
/* El tratamiento de creación de instancias tiene dos partes:                */
/*****************************************************************************/
/* Parte 1: leemos el fichero plantilla hasta encontrar null y lo            */
/* volcamos sobre el fichero completo, en el mismo orden de lectura          */
/*****************************************************************************/

      while ((sLine = brPlantillaOWL.readLine())!=null) {
        if (!sLine.equals("</Ontology>")) {
          bwDestinoOWL.write(sLine);
          bwDestinoOWL.newLine();
        }
      } /* while */

/*****************************************************************************/
/* Parte 2: creamos todas las instancias equivalentes a las posiciones       */
/* de fichero PGN de entrada                                                 */
/*****************************************************************************/

      indice = 0;
      Iterator it = games.keySet().iterator();
      bwDestinoOWL.write("<!-- Definicion de instancias-->");
      bwDestinoOWL.newLine();
      while (it.hasNext()) {
        indice += 1;
        String key = (String) it.next();
        Game gm = (Game) games.get(key);
/* Event: */
        bwDestinoOWL.write("<Declaration>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
        bwDestinoOWL.write("</Declaration>");
```

```
bwDestinoOWL.newLine();
bwDestinoOWL.write("<ClassAssertion>");
bwDestinoOWL.write("<Class IRI=\"#Event\"/>");
bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
bwDestinoOWL.write("</ClassAssertion>");
bwDestinoOWL.newLine();
if (gm.event != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventName\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.event + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.site != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#site\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.site + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.eventDate != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventDate\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eventDate + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.eventType != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventType\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eventType + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.eventCountry != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventCountry\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eventCountry + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.eventRounds != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventRounds\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eventRounds + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.eventSponsor != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eventSponsor\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eventSponsor + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.section != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#section\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.section + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
```

```
        if (gm.stage != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#stage\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.stage + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }

/* Player: */
        bwDestinoOWL.write("<Declaration>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
        bwDestinoOWL.write("</Declaration>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<ClassAssertion>");
        bwDestinoOWL.write("<Class IRI=\"#Player\"/>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
        bwDestinoOWL.write("</ClassAssertion>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<Declaration>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
        bwDestinoOWL.write("</Declaration>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<ClassAssertion>");
        bwDestinoOWL.write("<Class IRI=\"#Player\"/>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
        bwDestinoOWL.write("</ClassAssertion>");
        bwDestinoOWL.newLine();
        if (gm.white != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#name\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.white + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }
        if (gm.whiteTitle != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#title\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.whiteTitle + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }
        if (gm.whiteElo != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#elo\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.whiteElo + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }
        if (gm.whiteUSCF != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#uscf\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.whiteUSCF + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }
        if (gm.whiteNA != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
           bwDestinoOWL.write("<DataProperty IRI=\"#na\"/>");
           bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
           bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.whiteNA + "</Literal>");
           bwDestinoOWL.write("</DataPropertyAssertion>");
           bwDestinoOWL.newLine();
        }
        if (gm.whiteType != null) {
           bwDestinoOWL.write("<DataPropertyAssertion>");
```

```
            bwDestinoOWL.write("<DataProperty IRI=\"#typePlayer\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.whiteType + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.black != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#name\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.black + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.blackTitle != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#title\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.blackTitle + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.blackElo != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#elo\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.blackElo + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.blackUSCF != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#uscf\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.blackUSCF + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.blackNA != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#na\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.blackNA + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.blackType != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#typePlayer\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.blackType + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }

    /* Position o subclase: */
        bwDestinoOWL.write("<Declaration>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
        bwDestinoOWL.write("</Declaration>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<ClassAssertion>");
        bwDestinoOWL.write("<Class IRI=\"#" + sPosicion + "\"/>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
        bwDestinoOWL.write("</ClassAssertion>");
        bwDestinoOWL.newLine();
        if (gm.fen != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#fen\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.fen + "</Literal>");
```

```
        bwDestinoOWL.write("</DataPropertyAssertion>");
        bwDestinoOWL.newLine();
      }
      if (gm.move != null) {
        bwDestinoOWL.write("<DataPropertyAssertion>");
        bwDestinoOWL.write("<DataProperty IRI=\"#move\"/>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.move + "</Literal>");
        bwDestinoOWL.write("</DataPropertyAssertion>");
        bwDestinoOWL.newLine();
      }
      bwDestinoOWL.write("<DataPropertyAssertion>");
      bwDestinoOWL.write("<DataProperty IRI=\"#typePosition\"/>");
      bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");

      if     (sPosicion.equals("Position"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">undetermined</Literal>");
      else if (sPosicion.equals("PositionBegin"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">undetermined</Literal>");
      else if (sPosicion.equals("PositionMiddle"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">undetermined</Literal>");
      else if (sPosicion.equals("PositionEnd"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">undetermined</Literal>");
      else if (sPosicion.equals("OpeningError"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">strategy</Literal>");
      else if (sPosicion.equals("Trap"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">strategy</Literal>");
      else if (sPosicion.equals("Drowned"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("ContinuousCheck"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Combination"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("DoubleAttack"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("DiscoveredAttack"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("DiscoveredCheck"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("DoubleCheck"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Pinning"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Skewer"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Deflection"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Interception"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("EliminateDefense"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Freeing"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Lock"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Xray"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Overload"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("WeakLine"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Zwischenzug"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("Promotion"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("AttackKingCenter"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("AttackCastledKing"))
        bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">tactic</Literal>");
      else if (sPosicion.equals("PawnEndgame"))
```

```
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("RookEndgame"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("MinorPieceEndgame"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("QueenEndgame"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("OtherEndgame"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("MateInOne"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("MateInTwo"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("MateInThree"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         else if (sPosicion.equals("OtherMate"))
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">final</Literal>");
         bwDestinoOWL.write("</DataPropertyAssertion>");
         bwDestinoOWL.newLine();

         bwDestinoOWL.write("<ObjectPropertyAssertion>");
         bwDestinoOWL.write("<ObjectProperty IRI=\"#hasBoard\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#Board" + indice + "\"/>");
         bwDestinoOWL.write("</ObjectPropertyAssertion>");
         bwDestinoOWL.newLine();

         bwDestinoOWL.write("<ObjectPropertyAssertion>");
         bwDestinoOWL.write("<ObjectProperty IRI=\"#hasEvent\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#Event" + indice + "\"/>");
         bwDestinoOWL.write("</ObjectPropertyAssertion>");
         bwDestinoOWL.newLine();

         bwDestinoOWL.write("<ObjectPropertyAssertion>");
         bwDestinoOWL.write("<ObjectProperty IRI=\"#hasMatch\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
         bwDestinoOWL.write("</ObjectPropertyAssertion>");
         bwDestinoOWL.newLine();

         bwDestinoOWL.write("<ObjectPropertyAssertion>");
         bwDestinoOWL.write("<ObjectProperty IRI=\"#hasPlayerWhite\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerWhite" + indice + "\"/>");
         bwDestinoOWL.write("</ObjectPropertyAssertion>");
         bwDestinoOWL.newLine();

         bwDestinoOWL.write("<ObjectPropertyAssertion>");
         bwDestinoOWL.write("<ObjectProperty IRI=\"#hasPlayerBlack\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#" + sPosicion + indice + "\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#PlayerBlack" + indice + "\"/>");
         bwDestinoOWL.write("</ObjectPropertyAssertion>");
         bwDestinoOWL.newLine();

/* Board: */
         bwDestinoOWL.write("<Declaration>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#Board" + indice + "\"/>");
         bwDestinoOWL.write("</Declaration>");
         bwDestinoOWL.newLine();
         bwDestinoOWL.write("<ClassAssertion>");
         bwDestinoOWL.write("<Class IRI=\"#Board\"/>");
         bwDestinoOWL.write("<NamedIndividual IRI=\"#Board" + indice + "\"/>");
         bwDestinoOWL.write("</ClassAssertion>");
         bwDestinoOWL.newLine();
         if (gm.board != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#boardNum\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Board" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.board + "</Literal>");
```

```
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.mode != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#mode\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Board" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.mode + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }

/* Match: */
        bwDestinoOWL.write("<Declaration>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
        bwDestinoOWL.write("</Declaration>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<ClassAssertion>");
        bwDestinoOWL.write("<Class IRI=\"#Match\"/>");
        bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
        bwDestinoOWL.write("</ClassAssertion>");
        bwDestinoOWL.newLine();
        if (gm.date != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#date\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.date + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.round != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#round\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.round + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.result != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#result\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.result + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.opening != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#opening\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.opening + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.variation != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#variation\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.variation + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
        if (gm.subVariation != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#subVariation\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.subVariation + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
        }
```

```
if (gm.eco != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#eco\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.eco + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.nic != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#nic\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.nic + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.time != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#time\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.time + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.utcTime != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#utcTime\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.utcTime + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.utcDate != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#utcDate\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.utcDate + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.timeControl != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#timeControl\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.timeControl + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.setUp != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#setUp\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.setUp + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.termination != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#termination\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.termination + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
   bwDestinoOWL.newLine();
}
if (gm.annotator != null) {
   bwDestinoOWL.write("<DataPropertyAssertion>");
   bwDestinoOWL.write("<DataProperty IRI=\"#annotator\"/>");
   bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
   bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.annotator + "</Literal>");
   bwDestinoOWL.write("</DataPropertyAssertion>");
```

```
            bwDestinoOWL.newLine();
          }
          if (gm.plyCount != null) {
            bwDestinoOWL.write("<DataPropertyAssertion>");
            bwDestinoOWL.write("<DataProperty IRI=\"#plyCount\"/>");
            bwDestinoOWL.write("<NamedIndividual IRI=\"#Match" + indice + "\"/>");
            bwDestinoOWL.write("<Literal datatypeIRI=\"&rdf;PlainLiteral\">" + gm.plyCount + "</Literal>");
            bwDestinoOWL.write("</DataPropertyAssertion>");
            bwDestinoOWL.newLine();
          }
        } /* while */
        bwDestinoOWL.write("</Ontology>");
        bwDestinoOWL.newLine();
        bwDestinoOWL.write("<!-- Generated by GPAHS-->");
        bwDestinoOWL.newLine();

        System.out.println("Instancias OWL generadas ...: " + indice + " en el fichero " + sDestinoOWL);

/*****************************************************************************
Control de errores fichero OWL
*****************************************************************************/

      } catch(Exception e){
        e.printStackTrace();
      } finally {
// En el finally cerramos el fichero, para asegurarnos
// que se cierra tanto si todo va bien como si salta
// una excepcion.
        try {
          if( null != frPlantillaOWL ){
            frPlantillaOWL.close();
          }
          if( null != bwDestinoOWL ){
            bwDestinoOWL.flush();
            bwDestinoOWL.close();
          }
        } catch (Exception e2) {
          e2.printStackTrace();
        } /* catch */
      } /* finally */


/*****************************************************************************/
/* mostrar tabla de games si la traza está activa                            */
/*****************************************************************************/

      if (sTraza.equals("true")) {
        parser.games.ShowContenido();
      }

} /* main */

:};

////////////////////////////////////////////////////////////////////////////////
//TERMINALS                                                                   //
////////////////////////////////////////////////////////////////////////////////

terminal TK_CORCHETE_ABRIR, TK_CORCHETE_CERRAR;
terminal TK_ASTERISCO;
terminal TK_CADENASTRING, TK_STRING;
terminal TK_WIN, TK_LOST, TK_DRAW, TK_NAG;
terminal TK_EVENT, TK_SITE, TK_DATE, TK_ROUND, TK_WHITE, TK_BLACK, TK_RESULT;
terminal TK_WHITETITLE, TK_BLACKTITLE, TK_WHITEELO, TK_BLACKELO, TK_WHITEUSCF, TK_BLACKUSCF;
terminal TK_WHITENA, TK_BLACKNA, TK_WHITETYPE, TK_BLACKTYPE;
terminal TK_EVENTDATE, TK_EVENTTYPE, TK_EVENTCOUNTRY, TK_EVENTROUNDS, TK_EVENTSPONSOR;
terminal TK_SECTION, TK_STAGE, TK_BOARD, TK_OPENING;
terminal TK_VARIATION, TK_SUBVARIATION, TK_ECO, TK_NIC, TK_TIME, TK_UTCTIME, TK_UTCDATE;
terminal TK_TIMECONTROL, TK_SETUP, TK_FEN, TK_TERMINATION, TK_ANNOTATOR, TK_MODE, TK_PLYCOUNT;
terminal TK_EMAWHITE, TK_EMABLACK;
```

```
////////////////////////////////////////////////////////////////////////////////
//NON TERMINALS                                                               //
////////////////////////////////////////////////////////////////////////////////

non terminal PGN_database, PGN_game, tag_section, movetext_section, tag_pair;
non terminal pair_event, pair_site, pair_date, pair_round, pair_white, pair_black, pair_result;
non terminal pair_whiteTitle, pair_blackTitle, pair_whiteElo, pair_blackElo, pair_whiteUscf, pair_blackUscf;
non terminal pair_whiteNa, pair_blackNa, pair_whiteType, pair_blackType;
non terminal pair_eventDate, pair_eventType, pair_eventCountry, pair_eventRounds, pair_eventSponsor;
non terminal pair_section, pair_stage, pair_board, pair_opening;
non terminal pair_variation, pair_subVariation, pair_eco, pair_nic, pair_time, pair_utcTime, pair_utcDate;
non terminal pair_timeControl, pair_setUp, pair_fen, pair_termination, pair_annotator, pair_mode, pair_plyCount;
non terminal element, element_sequence, game_termination;
non terminal pair_emaWhite, pair_emaBlack;

////////////////////////////////////////////////////////////////////////////////
//GRAMMAR                                                                     //
////////////////////////////////////////////////////////////////////////////////

PGN_database::=
        | PGN_game PGN_database
        | error  {: parser.errors.add("Error en linea " + parser.scanner.getLine() + " - " + "Error de formato PGN"); :}
                ;
PGN_game::= tag_section movetext_section;

tag_section::=
        | tag_pair tag_section;

tag_pair::=     pair_event
        | pair_site
        | pair_date
        | pair_round
        | pair_white
        | pair_black
        | pair_result
        | pair_whiteTitle
        | pair_blackTitle
        | pair_whiteElo
        | pair_blackElo
        | pair_whiteUscf
        | pair_blackUscf
        | pair_whiteNa
        | pair_blackNa
        | pair_whiteType
        | pair_blackType
        | pair_eventDate
        | pair_eventType
        | pair_eventCountry
        | pair_eventRounds
        | pair_eventSponsor
        | pair_section
        | pair_stage
        | pair_board
        | pair_opening
        | pair_variation
        | pair_subVariation
        | pair_eco
        | pair_nic
        | pair_time
        | pair_utcTime
        | pair_utcDate
        | pair_timeControl
        | pair_setUp
        | pair_fen
        | pair_termination
        | pair_annotator
        | pair_mode
        | pair_plyCount
        | pair_emaWhite
```

```
            | pair_emaBlack
            | error  {: parser.errors.add("Error en linea " + parser.scanner.getLine() + " - " + "Identificador de etiqueta no definido"); :}
            ;

pair_event::=      TK_CORCHETE_ABRIR TK_EVENT TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.event = s.toString();
                :};
pair_site::=       TK_CORCHETE_ABRIR TK_SITE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.site = s.toString();
                :};
pair_date::=       TK_CORCHETE_ABRIR TK_DATE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.date = s.toString();
                :};
pair_round::=      TK_CORCHETE_ABRIR TK_ROUND TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.round = s.toString();
                :};
pair_white::=      TK_CORCHETE_ABRIR TK_WHITE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.white = s.toString();
                :};
pair_black::=      TK_CORCHETE_ABRIR TK_BLACK TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.black = s.toString();
                :};
pair_result::=     TK_CORCHETE_ABRIR TK_RESULT TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.result = s.toString();
                :};
pair_whiteTitle::=  TK_CORCHETE_ABRIR TK_WHITETITLE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.whiteTitle = s.toString();
                :};
pair_blackTitle::=  TK_CORCHETE_ABRIR TK_BLACKTITLE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.blackTitle = s.toString();
                :};
pair_whiteElo::=    TK_CORCHETE_ABRIR TK_WHITEELO TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.whiteElo = s.toString();
                :};
pair_blackElo::=    TK_CORCHETE_ABRIR TK_BLACKELO TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.blackElo = s.toString();
                :};
pair_whiteUscf::=   TK_CORCHETE_ABRIR TK_WHITEUSCF TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.whiteUSCF = s.toString();
                :};
pair_blackUscf::=   TK_CORCHETE_ABRIR TK_BLACKUSCF TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.blackUSCF = s.toString();
                :};
pair_whiteNa::=     TK_CORCHETE_ABRIR TK_WHITENA TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.whiteNA = s.toString();
                :};
pair_blackNa::=     TK_CORCHETE_ABRIR TK_BLACKNA TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.blackNA = s.toString();
                :};
pair_whiteType::=   TK_CORCHETE_ABRIR TK_WHITETYPE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
                parser.whiteType = s.toString();
                :};
pair_blackType::=   TK_CORCHETE_ABRIR TK_BLACKTYPE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                {:
```

```
            parser.blackType = s.toString();
            :};
pair_eventDate::=   TK_CORCHETE_ABRIR TK_EVENTDATE TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eventDate = s.toString();
            :};
pair_eventType::=   TK_CORCHETE_ABRIR TK_EVENTTYPE TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eventType = s.toString();
            :};
pair_eventCountry::= TK_CORCHETE_ABRIR TK_EVENTCOUNTRY TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eventCountry = s.toString();
            :};
pair_eventRounds::= TK_CORCHETE_ABRIR TK_EVENTROUNDS TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eventRounds = s.toString();
            :};
pair_eventSponsor::= TK_CORCHETE_ABRIR TK_EVENTSPONSOR TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eventSponsor = s.toString();
            :};
pair_section::=    TK_CORCHETE_ABRIR TK_SECTION TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.section = s.toString();
            :};
pair_stage::=      TK_CORCHETE_ABRIR TK_STAGE TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.stage = s.toString();
            :};
pair_board::=      TK_CORCHETE_ABRIR TK_BOARD TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.board = s.toString();
            :};
pair_opening::=    TK_CORCHETE_ABRIR TK_OPENING TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.opening = s.toString();
            :};
pair_variation::=   TK_CORCHETE_ABRIR TK_VARIATION TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.variation = s.toString();
            :};
pair_subVariation::= TK_CORCHETE_ABRIR TK_SUBVARIATION TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.subVariation = s.toString();
            :};
pair_eco::=       TK_CORCHETE_ABRIR TK_ECO TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.eco = s.toString();
            :};
pair_nic::=       TK_CORCHETE_ABRIR TK_NIC TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.nic = s.toString();
            :};
pair_time::=      TK_CORCHETE_ABRIR TK_TIME TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.time = s.toString();
            :};
pair_utcTime::=    TK_CORCHETE_ABRIR TK_UTCTIME TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.utcTime = s.toString();
            :};
pair_utcDate::=    TK_CORCHETE_ABRIR TK_UTCDATE TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.utcDate = s.toString();
            :};
pair_timeControl::= TK_CORCHETE_ABRIR TK_TIMECONTROL TK_CADENASTRING:s TK_CORCHETE_CERRAR
            {:
            parser.timeControl = s.toString();
            :};
```

```
pair_setUp::=      TK_CORCHETE_ABRIR TK_SETUP TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.setUp = s.toString();
                   :};
pair_fen::=        TK_CORCHETE_ABRIR TK_FEN TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.fen = s.toString();
                   :};
pair_termination::= TK_CORCHETE_ABRIR TK_TERMINATION TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.termination = s.toString();
                   :};
pair_annotator::=   TK_CORCHETE_ABRIR TK_ANNOTATOR TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.annotator = s.toString();
                   :};
pair_mode::=       TK_CORCHETE_ABRIR TK_MODE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.mode = s.toString();
                   :};
pair_plyCount::=    TK_CORCHETE_ABRIR TK_PLYCOUNT TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.plyCount = s.toString();
                   :};
pair_emaWhite::=    TK_CORCHETE_ABRIR TK_EMAWHITE TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.emaWhite = s.toString();
                   :};
pair_emaBlack::=    TK_CORCHETE_ABRIR TK_EMABLACK TK_CADENASTRING:s TK_CORCHETE_CERRAR
                   {:
                   parser.emaBlack = s.toString();
                   :};

movetext_section::= element_sequence game_termination
                  | game_termination;

element_sequence::= element element_sequence
                  | element
                  ;

element::= TK_STRING:s
                   {:
                   String temporal1 = s.toString();
                   if (parser.move == null) {
                      parser.move = temporal1;
                   } else {
                      String temporal2 = parser.move.concat(temporal1);
                      parser.move = temporal2;
                   }
                   :}
      | TK_NAG:n
                   {:
                   String temporal1 = n.toString();
                   parser.IncluirNAG(temporal1);

                   :}
                   ;

game_termination ::=      TK_WIN
                     {:
                     parser.id += 1;
                     parser.Semantica();
                     parser.CrearGame(parser.id);
                     parser.LimpiarGame();
                     :}
                    |   TK_LOST
                     {:
                     parser.id += 1;
                     parser.Semantica();
                     parser.CrearGame(parser.id);
```

```
parser.LimpiarGame();
:}
| TK_DRAW
{:
parser.id += 1;
parser.Semantica();
parser.CrearGame(parser.id);
parser.LimpiarGame();
:}
| TK_ASTERISCO
{:
parser.id += 1;
parser.Semantica();
parser.CrearGame(parser.id);
parser.LimpiarGame();
:}
;
```

5.2.4 Fichero PGN

A modo de ejemplo, en esta sección incluimos el contenido de un fichero PGN con pocas posiciones.

```
[Event "#1 Chess: 5334 Problems, Combinations, "]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in one"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "3q1rk1/5pbp/5Qp1/8/8/2B5/5PPP/6K1 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Qxg7# 1-0

[Event "#2 Chess: 5334 Problems, Combinations, "]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in one"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "2r2rk1/2q2p1p/6pQ/4P1N1/8/8/PPP5/2KR4 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Qxh7# 1-0

[Event "#3 Chess: 5334 Problems, Combinations, "]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in one"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "r2q1rk1/pp1p1p1p/5PpQ/8/4N3/8/PP3PPP/R5K1 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Qg7# 1-0

[Event "#4 Chess: 5334 Problems, Combinations, "]
[Site "Copyright 1994 Könemann"]
[Date "1994.??.??"]
[Round "?"]
[White "Mate in one"]
[Black "White to move"]
[Result "1-0"]
[SetUp "1"]
[FEN "6r1/7k/2p1pPp1/3p4/8/1R6/5PPP/5K2 w - - 0 1"]
[PlyCount "1"]
[EventDate "1994.??.??"]

1. Rh3# 1-0
```

5.2.5 Plantilla HTML

En esta sección incluimos la plantilla HTML de contenido estático.

```
<!DOCTYPE html>
<html lang="es">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>GPAHS</title>
  <link rel="stylesheet" href="css/style.css" type="text/css">

<!-- JavaScript de JQuery: -->
  <script type="text/javascript"
       charset="utf-8"
       src="js/jquery-1.5.2.js">
  </script>

<!-- JavaScript que genera el dinamismo de la pàgina, utilizando Jquery: -->
  <script type="text/javascript"
       charset="utf-8"
       src="js/GPAHS.js">
  </script>

<!-- JavaScript que genera el tablero: -->
  <script>

var fila = new Number();
var columna = new Number();
var col = new Number();
var pixelFila = new Number();
var pixelColumna = new Number();
var tamanyo = new Number();
var fenString = new String();
var listaCampos = new String();
var listaFilas = new String();
var caracter = new String();
var regEx1 = / /;
var regEx2 = /\//;

var bk = new Image(); // rey negro
var bq = new Image(); // dama negra
var br = new Image(); // torre negra
var bb = new Image(); // alfil negro
var bn = new Image(); // caballo negro
var bp = new Image(); // peon negro
var wk = new Image(); // rey blanco
var wq = new Image(); // dama blanca
var wr = new Image(); // torre blanca
var wb = new Image(); // alfil blanco
var wn = new Image(); // caballo blanco
var wp = new Image(); // peon blanco
// imagenes:
bk.src = "img/30/bk.png";
bq.src = "img/30/bq.png";
br.src = "img/30/br.png";
bb.src = "img/30/bb.png";
bn.src = "img/30/bn.png";
bp.src = "img/30/bp.png";
wk.src = "img/30/wk.png";
wq.src = "img/30/wq.png";
wr.src = "img/30/wr.png";
wb.src = "img/30/wb.png";
wn.src = "img/30/wn.png";
wp.src = "img/30/wp.png";

function init() {
// EJEMPLO DE ENTRADAS A GENERAR POR GPAHS:
// posicion 1:
```

```
//        var ctx1 = document.getElementById("canvas1").getContext('2d');
//        posicion(ctx1, "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq e7 0 1");
// posicion 2:
//        var ctx2 = document.getElementById("canvas2").getContext('2d');
//        posicion(ctx2, "r4b1r/pppbkBpp/q1n3n1/5p2/2NP4/1QP5/PP3PPP/RNB2RK1 w - - 0 1");
//<GPAHS_INSERT_1>
}

function posicion(ctx, fen) {

//tamaño de imagen:
tamanyo = 30;

//********************
// dibujar el tablero:
//********************
ctx.beginPath();
ctx.fillStyle = "rgb(204,204,153)";

ctx.fillRect(90,30,30,30); //fila 8
ctx.fillRect(150,30,30,30);
ctx.fillRect(210,30,30,30);
ctx.fillRect(270,30,30,30);
ctx.fillRect(60,60,30,30); //fila 7
ctx.fillRect(120,60,30,30);
ctx.fillRect(180,60,30,30);
ctx.fillRect(240,60,30,30);
ctx.fillRect(90,90,30,30); //fila 6
ctx.fillRect(150,90,30,30);
ctx.fillRect(210,90,30,30);
ctx.fillRect(270,90,30,30);
ctx.fillRect(60,120,30,30); //fila 5
ctx.fillRect(120,120,30,30);
ctx.fillRect(180,120,30,30);
ctx.fillRect(240,120,30,30);
ctx.fillRect(90,150,30,30); //fila 4
ctx.fillRect(150,150,30,30);
ctx.fillRect(210,150,30,30);
ctx.fillRect(270,150,30,30);
ctx.fillRect(60,180,30,30); //fila 3
ctx.fillRect(120,180,30,30);
ctx.fillRect(180,180,30,30);
ctx.fillRect(240,180,30,30);
ctx.fillRect(90,210,30,30); //fila 2
ctx.fillRect(150,210,30,30);
ctx.fillRect(210,210,30,30);
ctx.fillRect(270,210,30,30);
ctx.fillRect(60,240,30,30); //fila 1
ctx.fillRect(120,240,30,30);
ctx.fillRect(180,240,30,30);
ctx.fillRect(240,240,30,30);
ctx.fill();

// dibujar texto de filas y columnas:
ctx.beginPath();
ctx.fillStyle = "rgb(0,0,0)";
ctx.font = "10pt Verdana";
ctx.fillText("A",70,25);
ctx.fillText("B",100,25);
ctx.fillText("C",130,25);
ctx.fillText("D",160,25);
ctx.fillText("E",190,25);
ctx.fillText("F",220,25);
ctx.fillText("G",250,25);
ctx.fillText("H",280,25);
ctx.fillText("A",70,285);
ctx.fillText("B",100,285);
ctx.fillText("C",130,285);
ctx.fillText("D",160,285);
ctx.fillText("E",190,285);
```

```
ctx.fillText("F",220,285);
ctx.fillText("G",250,285);
ctx.fillText("H",280,285);
ctx.fillText("8",45,50);
ctx.fillText("7",45,80);
ctx.fillText("6",45,110);
ctx.fillText("5",45,140);
ctx.fillText("4",45,170);
ctx.fillText("3",45,200);
ctx.fillText("2",45,230);
ctx.fillText("1",45,260);
ctx.fillText("8",310,50);
ctx.fillText("7",310,80);
ctx.fillText("6",310,110);
ctx.fillText("5",310,140);
ctx.fillText("4",310,170);
ctx.fillText("3",310,200);
ctx.fillText("2",310,230);
ctx.fillText("1",310,260);
ctx.fill();

//**************************
//tratamiento del campo FEN:
//**************************
fenString = fen;
listaCampos = fenString.split(regEx1);

//************************************************
// Campo 1.- Posición de las piezas en el tablero:
//************************************************
listaFilas = listaCampos[0].split(regEx2);
for (i=0;i<=7;i++) {
   fila = 8 - i;
   columna = 1;
   while (listaFilas[i].length != 0) {
      caracter = listaFilas[i].substr(0,1);
      listaFilas[i] = listaFilas[i].slice(1);

      switch(caracter) {
      case "r":
         pixel(fila, columna);
         ctx.drawImage(br,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "n":
         pixel(fila, columna);
         ctx.drawImage(bn,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "b":
         pixel(fila, columna);
         ctx.drawImage(bb,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "q":
         pixel(fila, columna);
         ctx.drawImage(bq,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "k":
         pixel(fila, columna);
         ctx.drawImage(bk,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "p":
         pixel(fila, columna);
         ctx.drawImage(bp,pixelColumna, pixelFila,tamanyo,tamanyo);
         columna += 1;
         break;
      case "R":
```

```
        pixel(fila, columna);
        ctx.drawImage(wr,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      case "N":
        pixel(fila, columna);
        ctx.drawImage(wn,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      case "B":
        pixel(fila, columna);
        ctx.drawImage(wb,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      case "Q":
        pixel(fila, columna);
        ctx.drawImage(wq,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      case "K":
        pixel(fila, columna);
        ctx.drawImage(wk,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      case "P":
        pixel(fila, columna);
        ctx.drawImage(wp,pixelColumna, pixelFila,tamanyo,tamanyo);
        columna += 1;
        break;
      default:
        columna += parseInt(caracter);
        break;
      }
    }
}

//*************************
// Campo 2.- Turno de juego:
//*************************

ctx.beginPath();
if (listaCampos[1].search(/b/) == -1) {
// Turno de juego de blancas:
   ctx.fillStyle = "rgb(255,255,255)";
   ctx.strokeStyle = "rgb(0,0,0)"
   ctx.arc(360,255,8,0,rads(360),false);
} else {
// Turno de juego de negras:
   ctx.fillStyle = "rgb(0,0,0)";
   ctx.strokeStyle = "rgb(0,0,0)"
   ctx.arc(360,45,8,0,rads(360),false);
}
ctx.closePath();
ctx.fill();
ctx.stroke();


//*********************************
// Campo 3.- Posibilidad de enroque:
//*********************************

// enroque negras largo:
ctx.beginPath();
if (listaCampos[2].search(/q/) == -1) {
   ctx.fillStyle = "rgb(255,0,0)";
} else {
   ctx.fillStyle = "rgb(0,255,0)";
}
ctx.strokeStyle = "rgb(0,0,0)"
ctx.arc(30,45,4,0,rads(360),false);
```

```
ctx.closePath();
ctx.fill();
ctx.stroke();

// enroque negras corto:
ctx.beginPath();
if (listaCampos[2].search(/k/) == -1) {
   ctx.fillStyle = "rgb(255,0,0)";
} else {
   ctx.fillStyle = "rgb(0,255,0)";
}
ctx.strokeStyle = "rgb(0,0,0)"
ctx.arc(330,45,4,0,rads(360),false);
ctx.closePath();
ctx.fill();
ctx.stroke();

// enroque blancas largo:
ctx.beginPath();
if (listaCampos[2].search(/Q/) == -1) {
   ctx.fillStyle = "rgb(255,0,0)";
} else {
   ctx.fillStyle = "rgb(0,255,0)";
}
ctx.strokeStyle = "rgb(0,0,0)"
ctx.arc(30,255,4,0,rads(360),false);
ctx.closePath();
ctx.fill();
ctx.stroke();

// enroque blancas corto:
ctx.beginPath();
if (listaCampos[2].search(/K/) == -1) {
   ctx.fillStyle = "rgb(255,0,0)";
} else {
   ctx.fillStyle = "rgb(0,255,0)";
}
ctx.strokeStyle = "rgb(0,0,0)"
ctx.arc(330,255,4,0,rads(360),false);
ctx.closePath();
ctx.fill();
ctx.stroke();

//**************************
// Campo 4.- Captura al paso:
//**************************

if (listaCampos[3].search(/-/) == -1) {
// se puede capturar al paso
   ctx.beginPath();
   ctx.fillStyle = "rgb(204,204,153)";
   ctx.font = "Bold 8pt Verdana";
   ctx.fillText("C.P.",330,110);
   ctx.fillText(listaCampos[3],375,110);
   ctx.fill();
}

//**********************************
// Campo 5.- Movimientos sin captura:
//**********************************

ctx.beginPath();
ctx.fillStyle = "rgb(204,204,153)";
ctx.font = "Bold 8pt Verdana";
ctx.fillText("M.S.C.",330,140);
ctx.fillText(listaCampos[4],375,140);
ctx.fill();

//*********************************
// Campo 6.- Movimientos de partida:
```

```
//*********************************

ctx.beginPath();
ctx.fillStyle = "rgb(204,204,153)";
ctx.font = "Bold 8pt Verdana";
ctx.fillText("M.P.",330,170);
ctx.fillText(listaCampos[5],375,170);
ctx.fill();
}

function pixel(f, c) {
        pixelFila = 30 + ((8 - f) * 30);
        pixelColumna = 60 + ((c - 1) * 30);
}

function rads(x) { return Math.PI*x/180 }

   </script>

 </head>

 <body onLoad="init();">

 <header id="page_header">
    <table>
       <tr><td rowspan="2"><img src="img/chess.gif" alt="Logo GPAHS" /></td>
          <td ><h2>GPAHS</h2></td>
       </tr>
       <tr>
          <td><h3>Chess Exercises</h3></td>
       </tr>
    </table>
 </header>

 <div id="container">
<!-- EJEMPLO DE ENTRADAS A GENERAR POR GPAHS:                                    -->
<!--   <div class="posicion">                                                    -->
<!--      <p class="posicion_nombre">Posición 1</p>                              -->
<!--      <table border=".5">                                                    -->
<!--      <tr><td>                                                              -->
<!--      <p class="posicion_gráfico">                                          -->
<!--         <canvas id="canvas1" width="390" height="300">                     -->
<!--         Su browser no acepta elementos canvas del HTML5.                   -->
<!--         </canvas>                                                          -->
<!--      </p></td>                                                             -->
<!--      <td valign="top" align="right">Texto de la derecha, en el top</td>    -->
<!--      </table>                                                              -->
<!--      <p class="posicion_solucion">solución ...</p>                         -->
<!--      <ul class="actions">                                                  -->
<!--         <li class="solucion"><a href="" title="Solución">Solución</a></li> -->
<!--      </ul>                                                                 -->
<!--   </div> -->
<!--   <hr /> -->
<!--<GPAHS_INSERT_2>-->
 </div>

 <footer id="page_footer">
    <hr />
    <p>GPAHS - Generador de Posiciones de Ajedrez en HTML5 Semantico</p>
    <p>&copy; 2011 UOC</p>
 </footer>
 </body>

</html>
```

5.2.6 Ficheros de procesamiento por lotes

En esta sección se incluye el fuente de los ficheros de procesamiento por lotes que han sido utilizados para compilar y ejecutar los distintos ficheros PGN.

Fichero PFC_compilar.bat:

```
echo "Check your Java version ..."
java -version
pause

del lexGPAHS.java
del parser.java
del sym.java
pause


java -cp JLex_1.2.6.jar JLex.Main lexGPAHS.lex
pause

rename lexGPAHS.lex.java lexGPAHS.java
pause

java -cp CUP_10k.jar java_cup.Main cupGPAHS.cup
pause


javac -cp CUP_10k.jar;. -d . -Xlint:unchecked parser.java sym.java lexGPAHS.java
pause
```

Fichero PFC_ejecutar_unitaria.bat:

```
del .\html\Prueba1.html
del .\owl\Prueba1.owl
pause

java -cp CUP_10k.jar;. parser Prueba1.pgn GPAHSPlantilla.html ChessPosition.owl Position false
pause
```

Fichero PFC_ejecutar_verificacion.bat:

```
echo off
echo Borrado de ficheros destino
del .\html\Prueba1.html
del .\owl\Prueba1.owl
del .\html\Prueba2.html
del .\owl\Prueba2.owl
pause

echo.
echo Prueba 1: numero de argumentos incorrectos
echo.
java -cp CUP_10k.jar;. parser Combinations_1.pgn
echo.
pause

echo.
echo Prueba 2: extension de fichero fuente incorrecta
echo.
java -cp CUP_10k.jar;. parser ExtensionIncorrecta.png GPAHSPlantilla.html ChessPosition.owl Position true
echo.
pause
```

```
echo.
echo Prueba 3: fichero fuente no existe
echo.
java -cp CUP_10k.jar;. parser Noexiste.pgn GPAHSPlantilla.httl ChessPosition.owl Position true
echo.
pause

echo.
echo Prueba 4: extension plantilla HTML incorrecta
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.httl ChessPosition.owl Position true
echo.
pause

echo.
echo Prueba 5: plantilla HTML no existe
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantillaNoExiste.html ChessPosition.owl Position true
echo.
pause

echo.
echo Prueba 6: extension plantilla OWL incorrecta
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.html ChessPosition.pwl Position true
echo.
pause

echo.
echo Prueba 7: plantilla OWL no existe
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.html ChessPositionNoExiste.owl Position true
echo.
pause

echo.
echo Prueba 8: posicion tactica no valida
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.html ChessPosition.owl PositionNoValida true
echo.
pause

echo.
echo Prueba 9: tipo de traza no valido
echo.
java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.html ChessPosition.owl Position trueNoValido
echo.
pause

echo.
echo Prueba 10: activacion de la traza por consola
echo.
java -cp CUP_10k.jar;. parser Prueba1.pgn GPAHSPlantilla.html ChessPosition.owl Position true
echo.
pause

echo.
echo Prueba 11: conversion de caracteres no tratables por Protege
echo.
java -cp CUP_10k.jar;. parser Prueba2.pgn GPAHSPlantilla.html ChessPosition.owl Position true
echo.
pause
```

Fichero PFC_ejecutar_integrada.bat:

```
del .\html\200CombinacionesDeMatePolgar.html
del .\html\1001Reinfeld.html
```

```
del .\html\AnthologyofChessCombinations3.html
del .\html\MateInOne.html
del .\html\MateInTwo.html
del .\html\MateInThree.html
del .\html\OtherEndgame.html
del .\html\tactics.html
del .\html\TacticalAuerswald1.html
del .\html\TacticalAuerswald2.html
del .\html\Tactica1.html
del .\html\Tactica2.html
del .\html\Aperturas.html

del .\owl\200CombinacionesDeMatePolgar.owl
del .\owl\1001Reinfeld.owl
del .\owl\AnthologyofChessCombinations3.owl
del .\owl\MateInOne.owl
del .\owl\MateInTwo.owl
del .\owl\MateInThree.owl
del .\owl\OtherEndgame.owl
del .\owl\tactics.owl
del .\owl\TacticalAuerswald1.owl
del .\owl\TacticalAuerswald2.owl
del .\owl\Tactica1.owl
del .\owl\Tactica2.owl
del .\owl\Aperturas.owl

pause


java -cp CUP_10k.jar;. parser 200CombinacionesDeMatePolgar.pgn GPAHSPlantilla.html ChessPosition.owl Combination false
pause

java -cp CUP_10k.jar;. parser 1001Reinfeld.pgn GPAHSPlantilla.html ChessPosition.owl Combination false
pause

java -cp CUP_10k.jar;. parser AnthologyofChessCombinations3.pgn GPAHSPlantilla.html ChessPosition.owl Combination false
pause

java -cp CUP_10k.jar;. parser MateInOne.pgn GPAHSPlantilla.html ChessPosition.owl MateInOne false
pause

java -cp CUP_10k.jar;. parser MateInTwo.pgn GPAHSPlantilla.html ChessPosition.owl MateInTwo false
pause

java -cp CUP_10k.jar;. parser MateInThree.pgn GPAHSPlantilla.html ChessPosition.owl MateInThree false
pause

java -cp CUP_10k.jar;. parser Otherendgame.pgn GPAHSPlantilla.html ChessPosition.owl OtherEndgame false
pause

java -cp CUP_10k.jar;. parser tactics.pgn GPAHSPlantilla.html ChessPosition.owl Combination false
pause

java -cp CUP_10k.jar;. parser TacticalAuerswald1.pgn GPAHSPlantilla.html ChessPosition.owl PositionMiddle false
pause

java -cp CUP_10k.jar;. parser TacticalAuerswald2.pgn GPAHSPlantilla.html ChessPosition.owl PositionMiddle false
pause

java -cp CUP_10k.jar;. parser Tactica1.pgn GPAHSPlantilla.html ChessPosition.owl PositionMiddle false
pause

java -cp CUP_10k.jar;. parser Tactica2.pgn GPAHSPlantilla.html ChessPosition.owl PositionMiddle false
pause

java -cp CUP_10k.jar;. parser Aperturas.pgn GPAHSPlantilla.html ChessPosition.owl PositionBegin false
pause
```

5.2.7 Hoja de estilos CSS

En esta sección se incluye el contenido integro de la hoja de estilos utilizada en las páginas HTML generadas.

```
body{
  width: 960px;
  margin: 15px auto;
}

p{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:18px;
  font-weight:normal;
  color:rgb(0,0,0);
  line-height: 100%;
  text-align:left;
}

li{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:18px;
  font-weight:normal;
  color:rgb(204,204,153);
  line-height: 100%;
  text-align:left;
}

hr {
  color:rgb(204,204,153);
}


p.posicion_nombre{
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:20px;
  font-weight:bold;
  color:rgb(204,204,153);
  line-height: 150%;
  text-align:left;
  text-decoration:underline;
}

#data1 {
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:12px;
  font-weight:bold;
  color:rgb(0,0,0);
  text-align:left;
  vertical-align:top;
  }

#data2 {
  font-family: Verdana, "MS Trebuchet", sans-serif;
  font-style:normal;
  font-size:12px;
  font-weight:bold;
  color:rgb(255,255,255);
  background-color:rgb(204,204,153);
  text-align:left;
  vertical-align:top;
```

```
 }

h2{
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:25px;
 font-weight:bold;
 color:rgb(204,204,153);
 line-height: 20px;
 text-align:center;
 }

h3{
 font-family: Verdana, "MS Trebuchet", sans-serif;
 font-style:normal;
 font-size:25px;
 font-weight:bold;
 color:rgb(204,204,153);
 line-height: 20px;
 text-align:center;
 }
```

5.2.8 Ficheros JavaScript

En esta sección se incluye el código fuente del fichero GPAHS.js, responsable del dinamismo de la página HTML (con apoyo de la librería jquery-1.5.2.js).

```
// GPAHS - Generador de Posiciones de Ajedrez en HTML5 y contenido Semántico
// UOC - 2011

$(document).ready(function(){
  // ocultamos las soluciones de cada posición:
  $(".posicion .posicion_solucion").hide();


$(".actions li.solucion a").click(function(event){
    var $currElement = $("#start");
    $currElement = $(this).parents("ul");
    $currElement = $currElement.prev(".posicion_solucion");
    $currElement.toggle();
    event.preventDefault();
});

});
```

NOTA: *No se incluye el fuente del fichero jquery-1.5.2.js por contener más de 8000 líneas de código*.