SISTEMAS INTELIGENTES
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante **Curso 2015/2016**

Práctica 1. Satisfacción de restricciones

Objetivos:

- Aprender a definir un problema como un Problema de Satisfacción de Restricciones.
- Entender el funcionamiento e implementar los algoritmos backtracking y AC3.
- Aplicar este algoritmo a un problema concreto, en este caso, la resolución de un Sudoku.

Sesión 1. Introducción y entorno de trabajo. Diseño del algoritmo de búsqueda basada en backtracking.

En esta primera práctica de la asignatura se deben implementar los algoritmos backtracking y AC3 para la resolución de un sudoku.

1.1 Netbeans y Java

Para el desarrollo de la práctica se utilizará NetBeans y el lenguaje Java.

NetBeans es un entorno de desarrollo integrado, hecho principalmente para el lenguaje de programación Java, aunque extendido a otros lenguajes.

NetBeans es un proyecto de código abierto, por lo tanto, lo podemos descargar libremente desde su página oficial:

http://netbeans.org/

1.2 Enunciado

El sudoku es un juego en el que hay que rellenar una cuadrícula con números del 1 al 9. Se utiliza un tablero de 9x9, subdividido en 9 submatrices de 3x3.

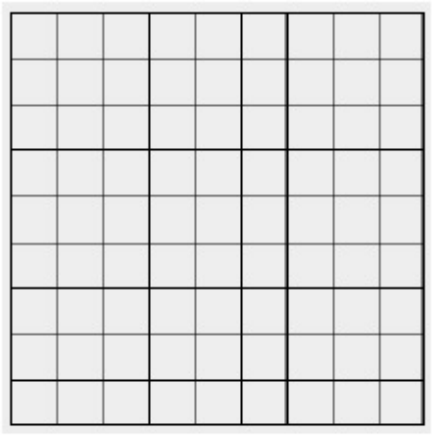


Imagen 1. Tablero de un Sudoku

En cada casilla del tablero se pueden colocar números del 1 al 9, con la restricción que ese número no puede repetirse en la misma fila, en la misma columna y en la submatriz donde se encuentre. Visto de otra manera, en cada fila deben aparecer los números del 1 al 9, en cada columna deben aparecer los números del 1 al 9 y cada submatriz debe tener los números del 1 al 9.

Para empezar el sudoku, se dan una serie de números iniciales y el resto de casillas se dejan vacías.

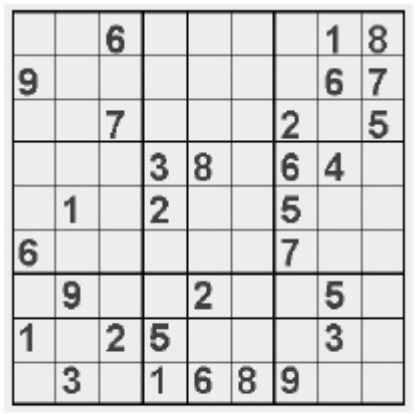


Imagen 2. Sudoku inicial

El sudoku está completo cuando se rellenan todas las casillas cumpliendo las reglas establecidas. Debemos ir colocando números del 1 al 9 en cada casilla vacía, de tal forma que no se repitan en su fila, columna o submatriz.

3	5	6	9	7	2	4	1	8
9	2	1	8	5	4	3	6	7
8	4	7	6	3	1	2	9	5
2	7	9	3	8	5	6	4	1
7	1	3	2	4	6	5	8	9
6	8	5	4	1	9	7	2	3
4	9	8	7	2	3	1	5	6
1	6	2	5	9	7	8	3	4
5	3	4	1	6	8	9	7	2

Imagen 3. Sudoku solucionado

Para la implementación del algoritmo backtracking para la resolución de un sudoku, se proporciona un entorno creado en NetBeans (puede verse en la imagen 4). Desde el menú `Archivo>>Cargar sudoku`, se puede cargar en el tablero un sudoku. Junto con el código de la práctica, están disponibles 4 sudokus de diferentes dificultades. Podéis generar vuestros propios sudokus creando un archivo de texto con el siguiente formato: (ejemplo Sudoku de la imagen 2).

0 0 6 0 0 0 0 1 8

9 0 0 0 0 0 0 6 7

0 0 7 0 0 0 2 0 5

0 0 0 3 8 0 6 4 0

0 1 0 2 0 0 5 0 0

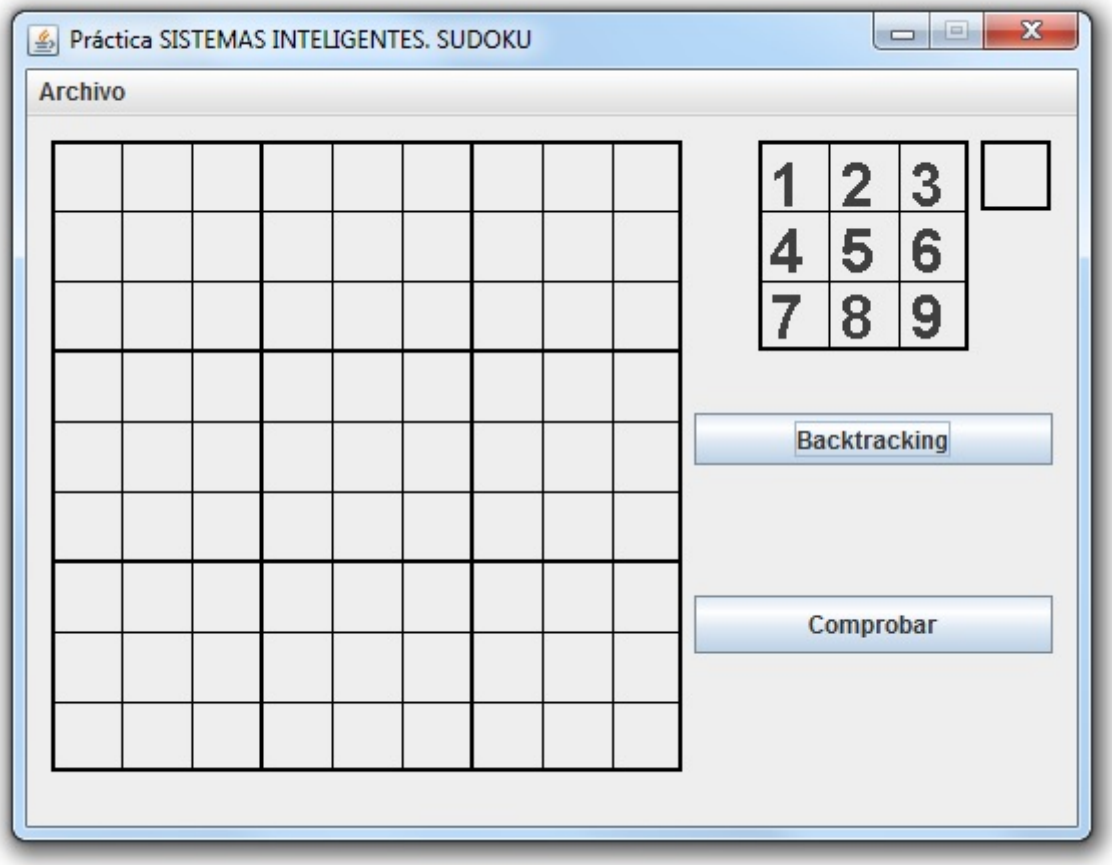
6 0 0 0 0 0 7 0 0

0 9 0 0 2 0 0 5 0

1 0 2 5 0 0 0 3 0

0 3 0 1 6 8 9 0 0

En cada fila se indican las nueve casillas, las casillas vacías se indican con un 0.



Utilizando los números de la parte derecha de la ventana también podemos crear directamente un sudoku, o añadir restricciones al que tenemos. Se debe seleccionar la casilla en la que se desea introducir el número y pulsar sobre el número.

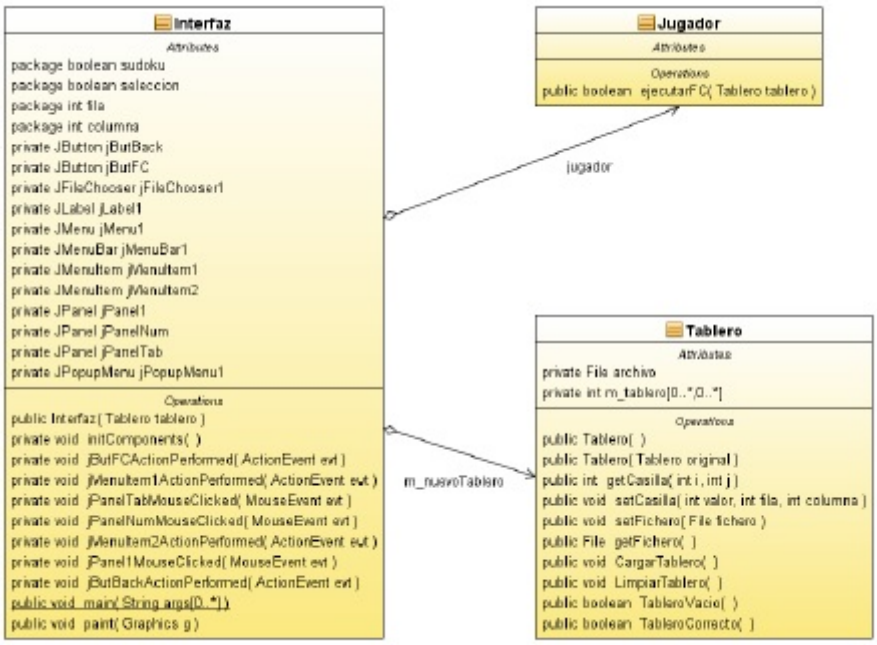
Al ejecutar el botón Backtracking se ejecutará el algoritmo, se mostrará en el tablero la solución del tablero y automáticamente mostrará si la solución es correcta o no.

El botón Comprobar, comprueba si un tablero es correcto.

Sesión 2. Implementación del algoritmo backtraking. Pruebas del algoritmo.

En esta sesión se debe realizar la implementación del algoritmo backtracking para resolver un sudoku.

Se proporciona un código base con el interfaz para el desarrollo de la práctica. ÚNICAMENTE se debe modificar el fichero Jugador. Se adjunta el diagrama UML del código adjunto:



Sesión 3. Diseño del algoritmo de búsqueda AC3

El algoritmo AC3 reduce los dominios de las variables eliminado aquellos valores que seguro no van a formar parte de la solución.

```
Q = {c(e_p) = <V_i, V_j>|e_p ∈ E, i ≠ j}
Mientras Q ≠ Ø hacer
      <V_k, V_m> = seleccionar_y_borrar(Q)
      cambio = falso
      Para todo v_k  ∈  D_k hacer
            Si no_consistente (v_k, D_m) entonces
                  borrar (v_k, D_k)
                  cambio = cierto
            FinSi
      FinPara
      Si Dk = Ø entonces salir_sin_solución FinSi
      Si cambio = cierto entonces
            Q = Q ∪ {c(e_r) = <V_i, V_k> | e_r ∈ E, i ≠ k, i ≠ m}
      FinSi
FinMientras
```

Sesión 4. Implementación del algoritmo de búsqueda AC3

Sesión dedicada a la implementación del algoritmo AC3

Sesión 5. Implementación del algoritmo de búsqueda AC3

Sesión dedicada a la implementación del algoritmo AC3

Sesión 6. Integración del sistema de búsqueda AC3. Elaboración de la documentación.

La documentación es una parte muy importante de la práctica. Debe incluir la explicación detallada de los algoritmos implementados

Sesión 7. Pruebas del algoritmo. Elaboración de la documentación.

La documentación deberá incluir una sección de experimentación en la que se describan las pruebas realizadas para diferentes dificultades de sudoku, dejando bien claro el objetivo de las pruebas, cómo se han llevado a cabo, qué información se ha recopilado a partir de las mismas, y qué conclusiones se han extraído. Una documentación sin este apartado se considerará suspensa.

Parte optativa. Implementación del algoritmo Forward Checking

Modificar el entorno gráfico de tal forma que aparezca otro botón que permita ejecutar el algoritmo Forward Checking.

Formato de entrega

La fecha límite de entrega es el 28 de octubre hasta las 12:00 de la noche. La entrega se realizará a través de Moodle. La entrega constará de un fichero .ZIP que contendrá dos carpetas:

- `'/src'` donde se encontrará la carpeta de proyecto de Netbeans.
- `'/doc'` donde estará disponible la documentación en formato PDF.

El nombre del fichero ZIP tendrá el siguiente formato: "`NombreApellido1Apellido2.ZIP`". Un fichero de ejemplo sería `RaulMartinezSerra.zip`

- ¡¡¡IMPORTANTE!!! no cumplir cualquiera de las normas de formato/entrega anteriores puede suponer un suspenso en la práctica. Recordad que las prácticas son individuales y NO se pueden hacer en parejas o grupos. Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia.
- ¡¡¡IMPORTANTE!!! únicamente se puede modificar la clase jugador, el resto de clases no se deben cambiar.