Práctica Inteligencia Artificial
Sistemas de producción en CLIPS
Resolución y generación de Sudokus

Ingeniería Técnica en Informática de Gestión
Curso 2006-07 (Febrero)

1. Introducción

En los últimos años el *Sudoku* se ha convertido en un pasatiempo muy popular. Tanto es así, que hoy en día numerosos periódicos publican Sudokus en su sección de pasatiempos. Parece ser que este rompecabezas numérico se originó en Nueva York hacia 1979, probablemente a partir de algunos trabajos del famoso matemático Leonhard Euler. Posteriormente, una editorial lo exportó a Japón, donde adoptó su nombre actual (su= número, doku= solo).

El Sudoku se presenta normalmente como una tabla 9×9, compuesta por subtablas de 3×3 denominadas *regiones* (también se llaman *cajas* o *bloques*). El objetivo es rellenar esta cuadrícula con cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas celdas (llamados *números dados* o *pistas*). La única restricción para rellenar las celdas vacías es que cada columna, fila y región debe contener los números del 1 al 9 sólo una vez (sin que ningún número esté repetido en ella). De ahí el nombre del juego: *los números deben estar solos*.

Desde un punto de vista informático, la resolución y generación automática de Sudokus se puede abordar de varias maneras. Por ejemplo, para resolver Sudokus, se podría plantear un algoritmo de *fuerza bruta* que fuera asignando números uno a uno a las celdas vacías y realizara un *backtracking* cada vez que se llegara a una situación en la que se viola alguna restricción. Sin embargo, esta solución aparte de ser poco eficiente computacionalmente, está lejos de la forma humana de razonar. Por lo general, cuando una persona se enfrenta a un Sudoku no utiliza un método de ensayo-error, sino que aplica una serie de reglas lógicas de tipo deductivo que le permiten ir rellenando las posiciones vacías. Utilizar un sistema de producción para implementar estas reglas parece bastante adecuado porque nos brinda la oportunidad de expresarlas de una manera directa y explícita, olvidándonos de alguna manera del flujo de control del programa. También puede ser de ayuda a la hora de generar Sudokus de distintos niveles de dificultad. Podríamos estimar la dificultad de resolver un Sudoku en función de la complejidad de las reglas que hay que aplicar para encontrar la solución.

La práctica consiste en diseñar e implementar en CLIPS un sistema de producción que sea capaz de resolver Sudokus y otro que sea capaz de generarlos. En esta práctica intentaremos emular la forma humana de razonar, por lo que únicamente trataremos Sudokus que se pueden resolver sin utilizar ninguna técnica de ensayo-error (es decir, en ningún caso trataremos de programar ningún tipo de *backtracking*).

2. Descripción

La realización de la práctica se dividirá en dos etapas. En la primera nos centraremos únicamente en la resolución de Sudokus mientras que en la segunda abordaremos su generación.

En cualquier caso, el proceso para construir el sistema de producción será el siguiente:

1. Modelizar el conocimiento para el sistema: definir las plantillas y/o la jerarquía de clases en CLIPS. Una vez definidas todas las clases con los campos adecuados, hay que hacer instancias concretas de esos objetos para generar la base de conocimiento.

2. Determinar las fases de las que el sistema se compone y construir "el algoritmo", es decir el conjunto de reglas que llevan a un funcionamiento correcto y adecuado.

2.1. Resolución de Sudokus

Para la resolución de Sudokus, la idea es comenzar analizando sistemáticamente cada celda vacía. Podemos empezar asumiendo que puede contener cualquier valor entre 1 y 9 y después eliminar todos los valores que ya estén asignados a otras celdas en su fila, columna o región. Esto permite conseguir un conjunto de candidatos para cada celda vacía.

Una vez que contamos con un conjunto de candidatos para cada celda vacía, se pueden aplicar una serie de reglas lógicas que permitan eliminar candidatos de manera que podamos conseguir un único candidato (que será el valor finalmente asignado a la celda). Para conseguir que el programa sea lo más sencillo y eficiente posible, se aplicarán antes las reglas de menor dificultad y en caso de que éstas no sean determinantes se pasará a aplicar reglas de mayor dificultad.

En la dirección web *http://www.lalunarosa.com/sudoku/* podéis encontrar una buena descripción de las reglas a aplicar ordenadas de menor a mayor dificultad. En cualquier caso podéis utilizar cualquier recurso web que os parezca adecuado.

Puesto que al terminar esta etapa no contaremos con un generador de Sudokus propio, también podéis utilizar cualquier generador que encontréis en la web para generar casos de prueba con los que evaluar el funcionamiento de vuestra práctica, como por ejemplo el que se encuentra en la página *http://kokolikoko.com/sudoku*.

2.2. Generación de Sudokus

Para la generación de Sudokus, también trataremos de emular la forma humana de razonar. No programaremos ningún método de ensayo-error (backtracking) en CLIPS. Una posible idea consiste en conseguir un Sudoku correcto y completo para a partir de él ir ocultando números de distintas posiciones.

Se puede conseguir un Sudoku completo y correcto partiendo de un Sudoku vacío en el que todos los números son candidatos de todas las casillas. Entonces, se puede elegir aleatoriamente un valor (de entre los candidatos) para una casilla, también escogida aleatoriamente. Cada vez que se fije un valor a una casilla, se eliminará el valor de todos los candidatos de todas las casillas en la fila, columna y región de la casilla asignada.

El único problema que presenta esta idea es que, casi con total seguridad, habrá casillas para las que eliminemos todos los candidatos, lo que supone que a estas casillas no se les podrá asignar un valor respetando las restricciones de juego. Sin embargo, podemos guiar la generación utilizando alguna regla heurística que sirva para que esto suceda lo menos posible. Una posible heurística consiste en seleccionar en cada paso la casilla que tiene menos candidatos (en vez de elegir la casilla aleatoriamente). Puesto que no programaremos ningún tipo de backtracking, si al finalizar el proceso hay casillas sin posibles candidatos, haremos otra iteración empezando desde el principio. Comprobaréis que de esta forma, en pocas iteraciones, se consigue un Sudoku completo y correcto.

Una vez que contemos con un Sudoku completo y correcto podremos ocultar números de varias casillas. La única condición que obligatoriamente debéis respetar es:

- Para considerar que un Sudoku generado es *válido*, deberá poder ser resuelto utilizando vuestro código para resolver Sudokus.

Curiosamente, la cantidad de números dados afecta a la dificultad del Sudoku menos de lo que parece, incluso puede no afectar en absoluto. Un Sudoku con un número mínimo de números dados puede ser muy fácil de resolver y uno con más números puede ser extremadamente complicado. La dificultad depende más de la relevancia y la posición de los números dados.

2.3. Formato de los ficheros de entrada/salida

Todos los Sudokus (tanto los casos de prueba para la parte de resolución, como los generados y sus soluciones) se almacenarán en ficheros de nombre libre que tendrán **obligatoriamente** el siguiente formato:

```
;; - - - | - 3 - | - 2 -
;; 5 2 3 | 9 - - | 1 6 -
;; 7 - - | 8 - 5 | 4 - -
;; ------+-------+------
;; 2 - 9 | 3 6 - | - - 1
;; - 4 7 | 1 - 8 | 2 3 -
;; - - - | - 7 2 | - - 9
;; ------+-------+------
;; - - 5 | - 8 9 | - - 6
;; - 9 1 | - - 6 | 7 - 2
;; - 6 - | - 1 - | - 5 -

(deffacts un-sudoku
   (casilla (fila 1) (columna 5) (valor 3))
   (casilla (fila 1) (columna 8) (valor 2))

   (casilla (fila 2) (columna 1) (valor 5))
   (casilla (fila 2) (columna 2) (valor 2))
   (casilla (fila 2) (columna 3) (valor 3))
   (casilla (fila 2) (columna 4) (valor 9))
   (casilla (fila 2) (columna 7) (valor 1))
   (casilla (fila 2) (columna 8) (valor 6))

   (casilla (fila 3) (columna 1) (valor 7))
   ...


)
```

Es decir, inicialmente se mostrará entre comentarios una representación gráfica del Sudoku que contiene el fichero. A continuación se expresarán los hechos que representan el contenido de todas de las casillas no vacías. Estos hechos deberán seguir una plantilla de nombre `casilla` con al menos tres slots (`fila`, `columna` y `valor`). Se tomará como fila 1 la fila superior y como columna 1 la que está a la izquierda del todo.

3. La competición

Cuando las dos etapas (resolución y generación) estén finalizadas, se realizará una mini-competición en la que cada grupo se enfrentará exclusivamente a otro grupo. Las parejas de grupos serán generadas aleatoriamente por los profesores.

A través de la competición, cada grupo podrá obtener hasta 0.5 puntos adicionales a los 4 puntos sobre los que se valoran las prácticas (es decir, en este caso se podrá alcanzar una puntuación de hasta 4.5 puntos).

El proceso para llevar a cabo la competición es el siguiente:

- Se reunirán los dos grupos a competir.
- Cada grupo deberá generar hasta 10 Sudokus que sean, al mismo tiempo, resolubles por su propio código.
- El grupo contrario tratará de resolver estos 10 Sudokus.
- El cálculo de la puntuación final de un grupo se realizará como: $0,5 \times \frac{(\mathrm{ng}+\mathrm{nr})}{20}$, donde:
  - ng es el número de Sudokus generado por el grupo que el grupo contrario no fue capaz de resolver.

- nr es el número de Sudokus generado por el grupo contrario que el grupo ha resuelto.

- La competición se realizará preferiblemente en el horario de clase la semana del 9 de enero.
- Como resultado de la competición, se entregará un acta firmada por los 4 miembros de los dos grupos que compiten, en el que aparezcan los 20 Sudokus (10 generados por cada grupo) con los que se realizó la competición y los resultados de la misma, incluyendo las soluciones a los Sudokus resueltos y el tiempo que se tardó en resolver cada uno.
- Los grupos que deseen participar en la competición deberán enviar un **único** e-mail con subject `competición iat` a Raquel Fuentetaja (rfuentet@inf.uc3m.es) en Leganés y a Manuel González (mgbedia@inf.uc3m.es) en Colmenarejo. El último día para mandar este correo es el 8 de enero de 2007 a las 15:00.

4. Se pide

1. Definir las clases, plantillas, instancias y hechos generales que sean necesarios, en el fichero **ontologia.clp**
2. Implementar las reglas necesarias para la resolución de Sudokus, en el fichero **resolucion.clp**
3. Realizar una evaluación del funcionamiento del programa utilizando 10 Sukodus con distintos niveles de dificultad. Para esto se utilizará un fichero por cada uno de los 10 Sudokus utilizados en las pruebas con nombre **sudoku-i.clp**, donde i representa el número de prueba. Sus soluciones (si vuestro sistema las encuentra) se almacenarán en un fichero con nombre **solucion-sudoku-i.clp**.
4. Implementar las reglas necesarias para la generación de Sudokus, en el fichero **generacion.clp**

   Las reglas que se definan en ambos casos deben ser cortas y procurando siempre que optimice la velocidad del sistema.
5. Hacer una traza de las acciones que se van realizando en cada momento. Deberéis entregar una traza del programa que resuelve Sudokus y otra del generador. (Con el comando *dribble-on "fichero"* se consigue que todos los *printout* del programa se salven en un fichero).
6. En caso de realizar la competición, realizar un informe con los resultados de la misma.

5. Recomendaciones

- En general, se deberán representar los hechos utilizando plantillas (templates) o clases, salvo casos excepcionales en los que esté muy claro que no es necesario.
- No se deben usar **ifs** en la parte derecha de la regla o **tests** muy cargados en la parte izquierda.
- Se debe evitar el uso de funciones y variables globales. En caso de estimar necesario su uso consultad con alguno de los profesores de prácticas.
- Procurad diseñar el sistema basado en reglas de forma que las reglas sean escuetas con un objetivo sencillo y claro, y fáciles de comprender, evitando al mismo tiempo la proliferación de reglas demasiado simples. Se deben utilizar plantillas que representen conocimiento, de forma que las reglas sean lo más generales posibles
- El agrupamiento de varias reglas en una sola utilizando OR no significa que la regla obtenida esté simplificada. Es preferible construir varias reglas porque facilita la comprensión.
- Probablemente los test de igualdad serán innecesarios.
- El código deberá ser lo más legible posible, incluyendo la indentación y los comentarios adecuados.

- La documentación sobre CLIPS se puede encontrar en:
  $http://scalab.uc3m.es/\ docweb/ia/transparencias.html$
- El manual de referencia de CLIPS, en html:
  $http://scalab.uc3m.es/\ docweb/ia/CLIPS\_REFERENCIA$
- Las diferentes versiones de CLIPS se pueden encontrar en:
  $http://scalab.uc3m.es/\ docweb/ia/software.html$

6. Entrega de la práctica

Habrá una entrega principal (3 puntos) y una mini-entrega (0,25 puntos). La mini-entrega servirá para que los profesores os confirmen que la práctica está bien orientada.

La mini-entrega se realizará como muy tarde el día **1 de diciembre de 2006** en los casilleros de los profesores de prácticas de la asignatura. Se podrán entregar un máximo de 5 páginas explicando las plantillas, clases, algún ejemplo de reglas y cualquier consideración de diseño que os parezca relevante.

La entrega principal de la práctica se realizará como muy tarde el día **15 de enero de 2007**. Habrá que entregar una memoria en papel y los ficheros correspondientes a la práctica cuya entrega se realizará por Aula Global. **Es muy importante seguir estrictamente las normas de entrega aquí descritas**.

La memoria se entregará directamente a los profesores de práticas o se dejará en uno de sus casilleros.

- La memoria debe tener la siguiente estructura:

  1. Portada con los nombres y direcciones de correo de los autores.
  2. Introducción. Una hoja de descripción de la práctica desde vuestro punto de vista, no desde la copia del enunciado
  3. **Código y explicación** de:
     - Estructura estática de la memoria de trabajo (plantillas y jerarquía de marcos).
     - Las reglas que representan las inferencias del sistema de producción para la fase de resolución. Se debe incluir una explicación general sobre cómo están organizadas estas reglas (por ejemplo, si hay varias reglas que realizan una determinada función).
     - Las reglas que representan las inferencias del sistema de producción para la fase de generación. Se debe incluir una explicación general sobre cómo están organizadas estas reglas (por ejemplo, si hay varias reglas que realizan una determinada función).
  4. Evaluación de la parte correspondiente a la resolución.
  5. Conclusiones (resumen de lo realizado y comentarios generales acerca de la práctica y el uso de sistemas de producción, dificultades encontradas, etc.)

- Los ficheros se entregarán por Aula Global. Se entregará un único fichero comprimido (*.tgz*) cuyo nombre será el NIA de uno de los autores precedido por la letra *a* . Al descomprimir este fichero se deberá generar un directorio cuyo nombre debe corresponder con el NIA de uno de los autores precedido por la letra *a*. Este directorio debe contener exclusivamente los siguientes ficheros con los **nombres que se especifican**

  - Fichero con el nombre y la dirección de e-mail de los autores, **autores.txt**
  - El fichero con las definiciones de clases, plantillas, hechos e instancias generales, **ontologia.clp**
  - El fichero con las reglas para resolver Sudokus **resolucion.clp**
  - Un fichero por cada uno de los 10 Sudokus utilizados en las pruebas con nombre **sudoku-i.clp**, donde i representa el número de prueba. Por cada uno de estos ficheros, si vuestro código ha sido capaz de resolverlo, la solución a cada Sudoku, expresada en el formato definido anteriormente, en un fichero con nombre **solucion-sudoku-i.clp**.

- Los ficheros de traza, **traza-resolucion.txt** y **traza-generacion.txt**, de algún ejemplo de ejecución del programa para ambos casos.

Ejemplo: supongamos que el NIA de uno de los autores es 100000000. Por lo tanto, los ficheros a entregar estarán en un directorio con nombre a100000000. Para generar el fichero comprimido que se debe entregar, haremos:

tar -czvf a100000000.tgz a100000000/

La práctica se debe hacer en grupos de dos personas. En ningún caso se admitirán prácticas de grupos con más de dos alumnos