CURSO ACADÉMICO 2012-2013 - INGENIERÍA INFORMÁTICA (PLAN 96)
ASIGNATURA: MODELOS DE RAZONAMIENTO

PLANIFICACIÓN CON REDES DE TAREAS JERÁRQUICAS Y LA HERRAMIENTA JSHOP2

Profesor: Miguel García Remesal

PLANIFICACIÓN HTN USANDO LA HERRAMIENTA JSHOP2

En esta práctica se repasarán los conceptos básicos de planificación automática vistos en clase, y más concretamente se hará hincapié en la planificación basada en redes de tareas jerárquicas (planificadores HTN o "Hierarchical Task Network"). Para ello se utilizará la herramienta JSHOP2. Esta herramienta es un compilador de planificadores HTN desarrollada en Java, que permite, dada la descripción de un dominio de planificación HTN (que contiene tanto operadores primitivos como métodos de descomposición), generar automáticamente un programa Java que implemente el planificador asociado al dominio especificado. Dicho planificador puede utilizarse posteriormente para resolver cualquier problema de planificación asociado al citado dominio.

La herramienta JSHOP2 debe descargarse haciendo click sobre el siguiente enlace:

http://sourceforge.net/projects/shop/files/JSHOP2/JSHOP2GUI%201.0.1/JSHOP2GUI_1.0.1.zip/download

Una vez descargada la herramienta, el alumno dispondrá del archivo JSHOP2GUI_1.0.1.zip. Este archivo debe descomprimirse. Tras la descompresión del archivo, este generará la carpeta JSHOP2GUI_1.0.1, que contiene todos los archivos asociados a la herramienta. Los ficheros y carpetas más importantes son los siguientes:

- **Carpeta bin.build**, que contiene el archivo JSHOP2.jar, que contiene el bytecode de todas las clases Java asociadas al compilador de planificadores HTN.

- **Carpeta src**, que contiene el código fuente de la herramienta JSHOP2 (ya que esta es open-source).

- **Carpeta examples,** que contiene varios ejemplos de ficheros de definición de dominios y de problemas.

- **Carpeta doc**, que contiene la documentación del API asociada a la herramienta JSHOP2, ya que es posible llamar a esta herramienta desde programas externos, y no solamente utilizarla como una aplicación "stand alone".

- **Archivo JSHOP2.pdf**, que contiene una extensa documentación sobre la herramienta JSHOP2 (en inglés).

- **Archivo antlr.jar**, que contiene el bytecode de todas las clases Java asociadas a la herramienta antlr, que es un generador automático de "parsers" o analizadores sintácticos. Esta herramienta es necesaria para poder hacer el "parsing" de los ficheros de definición de dominio y de problema asociados al planificador, ya que estos están definidos en un lenguaje formal similar al utilizado en los lenguajes de programación de alto nivel.

Para poner en funcionamiento la herramienta es necesario seguir los siguientes pasos:

- Instalar el Java Development Kit (JDK) de Java (puede descargarse de la página web de Oracle). No basta con instalar el entorno de ejecución Java (JRE), ya que necesitaremos hacer uso del compilador.

- Añadir la ruta completa de los ficheros **bin.buid/JSHOP2.jar** y **antlr.jar** a la variable de entorno CLASSPATH. Esto debe hacerse tecleando la siguiente orden desde el terminal (sistemas Linux/Windows/OS X) o desde la línea de comandos CMD (en sistemas Windows

):

- **export CLASSPATH=$CLASSPATH:$JSHOP2_HOME/bin.build/JSHOP2.jar: $JSHOP2_HOME/antlr.jar** (en entornos Linux/Unix/OSX)
- **set CLASSPATH=%CLASSPATH%;%JSHOP2_HOME%\bin\build\JSHOP2.jar; %JSHOP2_HOME%\antlr.jar** (en entornos Windows)
- **Nota:** para que estos comandos surtan efecto, es necesario declarar la variable de entorno JSHOP2_HOME, que deberá contener la ruta completa al directorio JSHOP2GUI_1.0.1.

Una vez hecho esto, el alumno dispone ya de la herramienta lista para su uso. Se recomienda crear un pequeño "shell script" que realice estos pasos anteriores de manera automática, ya que es necesario hacerlos cada vez que se reinicie el ordenador y/o se cierre la ventana del terminal o línea de comandos. El alumno puede también probar el correcto funcionamiento de la herramienta compilando y ejecutando alguno de los planificadores de ejemplo que pueden encontrarse en el directorio **examples**. Para ello, deberá hacer lo siguiente:

- En este ejemplo se compilará el planificador "freecell", que puede encontrarse en la carpeta **examples/freecell**. El primer paso sería por tanto, cambiar el directorio actual a la citada carpeta.

- Dentro de esta carpeta, nos encontramos con diferentes ficheros. El fichero **freecell** contiene la definición del dominio de planificación. La primera tarea a realizar sería compilar dicho dominio, lo cual puede hacerse ejecutando el siguiente comando:

  java JSHOP2.InternalDomain freecell

  Esto nos genera varios ficheros, siendo el más importante el archivo **freecell.java**, que contiene el código Java que implementa el planificador HTN asociado al domino descrito en el fichero **freecell**.

- Dado que lo que se ha generado mediante el comando anterior es un programa Java, para poderlo ejecutar es necesario obtener el bytecode asociado al planificador (es decir, los archivos .class ejecutables por la máquina virtual Java). Para ello, compilamos el fichero generado en el paso anterior ejecutando el siguiente comando:

  javac freecell.java

  Lo cual genera una serie de archivos .class, que son los "ejecutables" asociados al planificador.

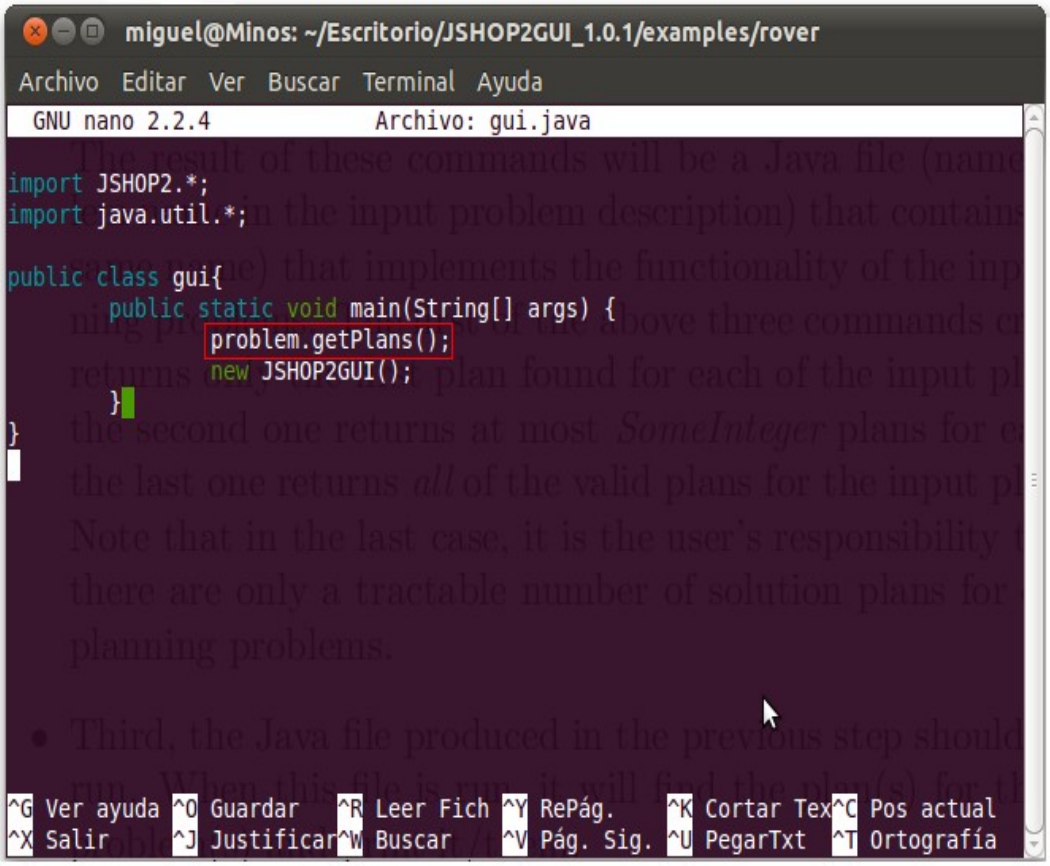
- Una vez generados estos archivos, tenemos ya compilado y listo para ejecutar el planificador correspondiente. Ahora sería necesario especificar y compilar un problema de planificación asociado al dominio anterior. Uno de estos problemas es el que podemos encontrar en el fichero **problem**. Para compilar este problema, será necesario ejecutar el siguiente comando:

  java JSHOP2.InternalDomain -r problem

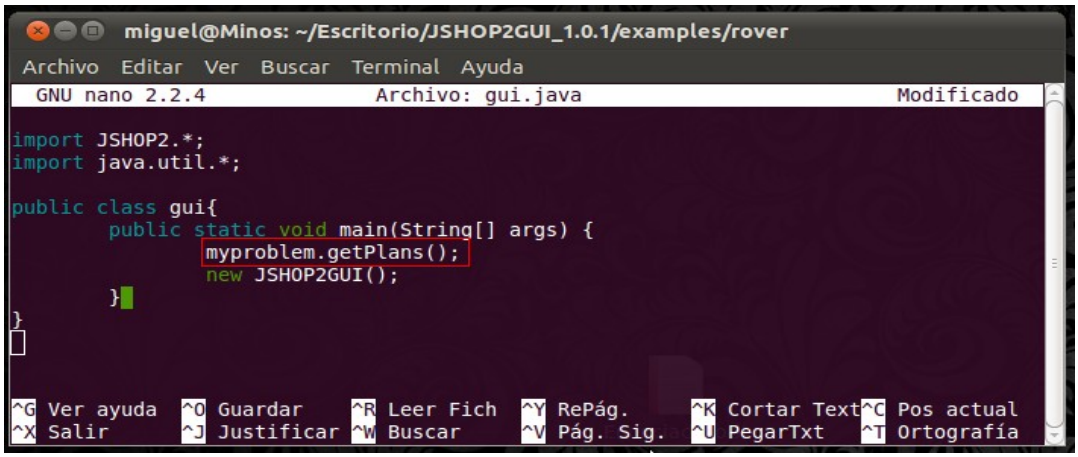
  que genera el archivo problem.java, que es el código fuente Java del problema de planificación definido en el archivo **problem**.

- Dado que JSHOP2 dispone de una interfaz gráfica que permite ir ejecutando el planificador paso a paso, será necesario compilar el programa Java **gui.java**, que puede encontrarse

también en el directorio **examples/freecell**. Este programa se apoya en el fichero Java recién generado “problem.java” para construir la interfaz gráfica, tal y como puede verse en la siguiente figura:



Tal y como se indica en la figura, es necesario llamar al método getPlans() de la clase problem, que viene implementada en el fichero **problem,java**, generado automáticamente en el paso anterior. Por tanto, esto significa que si el alumno define un problema diferente con un nombre distinto, este deberá modificar el código del fichero **gui.java** para que el programa GUI resuelva el problema definido por el alumno, en lugar del problema original “problem”. Por ejemplo, si se supone que el alumno ha definido el problema “myproblem”, y ha obtenido el fichero java **myproblem.java** tras la compilación de dicho problema, entonces el código del programa **gui.java** deberá modificarse de la manera siguiente:

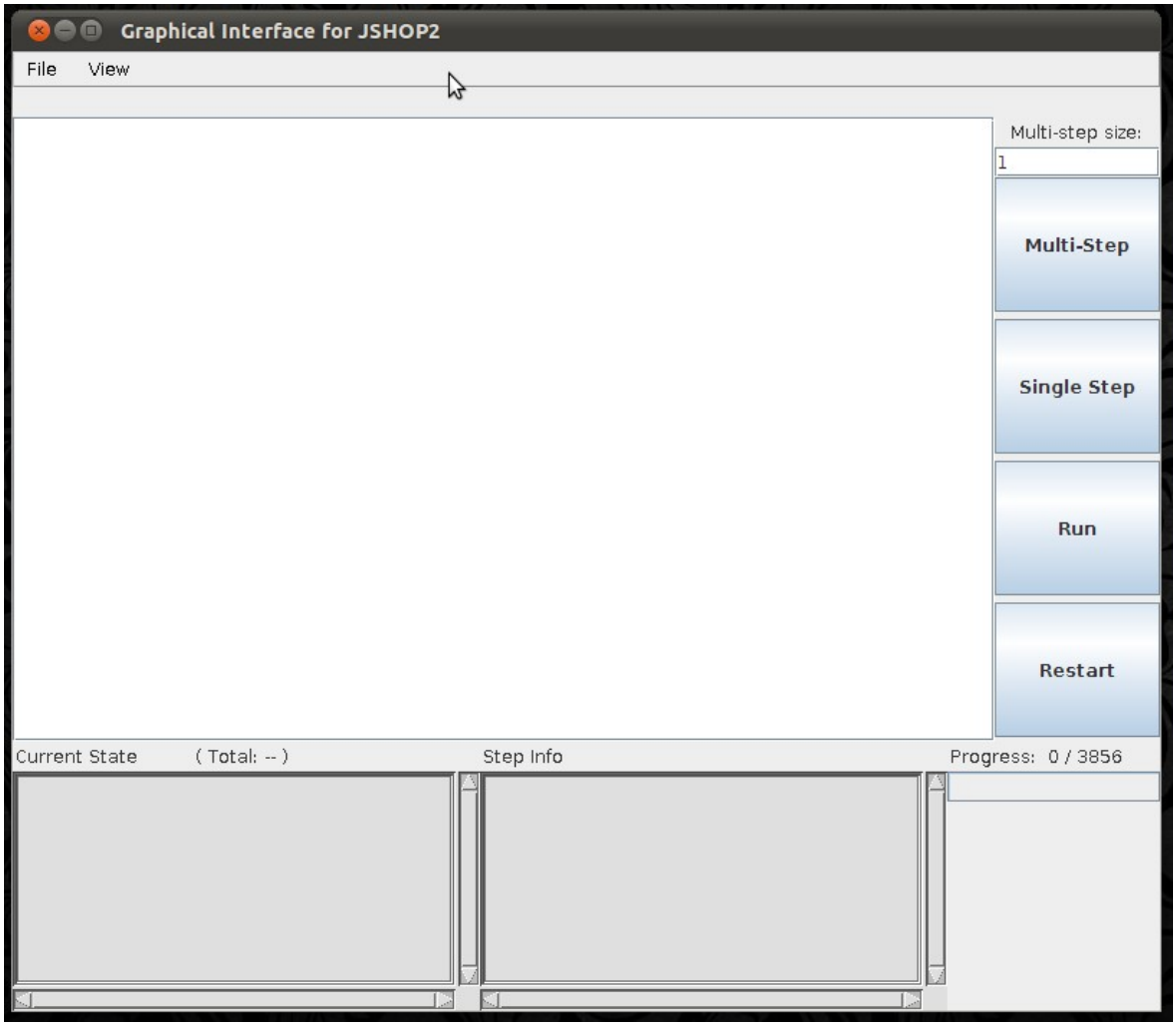


- Una vez realizadas las modificaciones pertinentes en el fichero **gui.java**, basta con compilarlo con el comando siguiente:

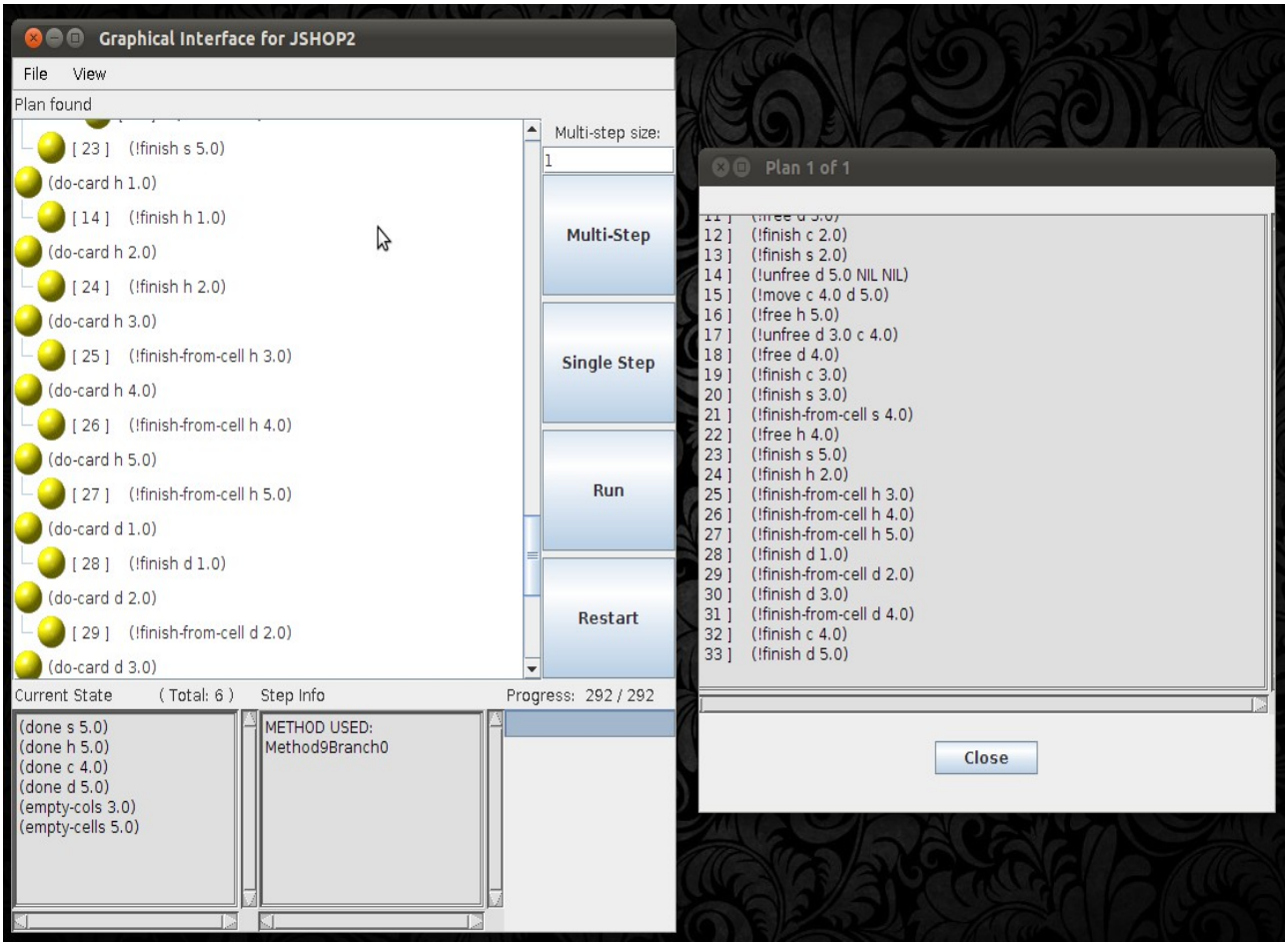
  javac gui.java

  Tras lo cual se obtiene el fichero **gui.jar**, que contiene interfaz gráfico para interactuar con el planificador asociado al problema concreto definido por el alumno.

- Para ver el interfaz del planificador, basta con teclear el comando: **java gui**, lo cual lleva al alumno a la pantalla siguiente:



Como se puede observar, hay varios botones. El botón “multi-step” sirve para ir viendo la ejecución del planificador paso a paso. Es posible variar el tamaño del paso al valor que se desee, introduciendo para ello dicho valor en el campo “Multi-step size”. Como puede verse, el tamaño por defecto del paso es 1. Esta opción es de gran utilidad para tareas de depuración. Por otra parte, el botón “single step” sirve para ir viendo la ejecución paso a paso. Asimismo, el botón “Run” sirve para obtener el plan completo (útil cuando no se está depurando), y el botón restart vale para reiniciar el planificador. Si el alumno pulsa el botón “Run”, obtendrá el plan solución del problema, si es que este existe. La figura siguiente muestra la pantalla donde se expone la solución (plan) encontrado por el planificador para el problema propuesto.



Como se puede ver en la figura anterior, el planificador encuentra una solución, que está compuesta por las 33 acciones que pueden verse en la ventana de la derecha. La ventana de la izquierda, por contra, muestra la descomposición realizada por el planificador HTN construido mediante la herramienta JSHOP2.

En lo que se refiere a la práctica en sí, se pide:

- Seleccionar un problema de complejidad media-alta que sea adecuado para su resolución mediante técnicas de planficación. Con complejidad media-alta estamos hablando de un problema **que necesite al menos de 4 operadores primitivos**.
- Formalizar el dominio descrito en el enunciado utilizando el formalismo clásico STRIPS. Explicar CLARAMENTE el significado de cada constante, predicado y operador utilizado.
- Utilizando conocimiento de dominio obtenido vía entrevista con un experto en el dominio del problema, o bien mediante búsqueda de información en libros u otras fuentes, etc., redefinir el dominio básico como un dominio de planificación HTN. El dominio formalizado debe contener **al menos 4 métodos de descomposición**.
- Implementar un planificador HTN que permita resolver problemas en el dominio de planificación anterior utilizando la herramienta JSHOP2.
- Definir y resolver al menos dos problemas de planficación diferentes utilizando el planificador implementado.
- Comentar y discutir brevemente los resultados obtenidos. Reflejar ventajas e inconvenientes de este tipo de planificadores frente a otros planificadores más sencillos.

NORMAS DE PRESENTACIÓN

Se deberá entregar una memoria EN PAPEL de no más de 15 páginas, describiendo tanto el desarrollo de la práctica como los resultados obtenidos. También es necesario incluir un breve comentario personal sobre la práctica, indicando además el número de horas de trabajo que le ha llevado al alumno la realización de la misma. La memoria impresa puede entregarse personalmente al profesor Miguel García Remesal (despacho D-2206) o bien puede dejarse a su nombre en la secretaría del DIA. Será necesario también enviar una copia de la memoria en formato PDF junto con todos los archivos necesarios para reproducir los resultados descritos en la memoria por correo electrónico a la dirección mgremesal@fi.upm.es. El correo electrónico DEBE TENER el siguiente asunto: "PRACTICA MR 1213". Todos los archivos (incluyendo el archivo PDF con la memoria) deben empaquetarse dentro de un archivo RAR o ZIP, el cual debe nombrarse según el siguiente formato:

1011-MR-[APELLIDOS_NOMBRE].RAR

Los nombres compuestos y apellidos múltiples deben unirse mediante un guión (-).

Ejemplos:

Martin Wheeler: 1213-MR-[Wheeler_Martin].rar
Marcos Ángel Martínez Díaz: 1213-MR-[Martinez-Diaz_Marcos-Angel].zip

Composición del grupo de prácticas

La práctica debe realizarse de forma individual.

¡IMPORTANTE!

Tanto las normas de nombrado como el asunto del e-mail deben seguirse al pie de la letra, ya que de otro modo, no puede garantizarse la recepción correcta de la práctica.

Fecha tope de entrega:

La práctica debe entregarse antes de las 13:00 horas del 13 de Junio de 2013.