Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 1 – Rush Hour Extreme
IIC2613 - Inteligencia Artificial
Segundo Semestre, 2015
Entrega: 6 de Octubre, hasta las 23:59

1. Objetivo

El objetivo de esta tarea es doble. En primer lugar, practicará el uso de un lenguaje de programación basado en lógica (Prolog); en segundo, se espera que usted se familiarice con el algoritmo A* y comprenda el impacto en su rendimiento al ser usado con diferentes funciones heurísticas.

2. Rush Hour Extreme

Rush Hour se juega en una cuadrícula de 6×6, moviendo vehículos que están dispuestos en orientación ya sea vertical u horizontal. El problema siempre consiste en sacar el vehículo rojo fuera de la cuadrícula. Para llevarse una precisa impresión sobre el juego, diríjase a la página `http://www.thinkfun.com/mathcounts/play-rush-hour` y juéguelo.

Rush Hour Extreme (RHE) es una variante de Rush Hour donde los vehículos no se pueden mover hasta una posición arbitraria sino que se desplazarán lo más lejos posible en la dirección que son desplazados. Se puede entender, entonces, que para cada vehículo V dispuesto horizontalmente existen solo dos acciones disponibles $right(V)$ y $left(V)$. La acción $right(V)$ es posible solo si el cuadro inmediatamente a la derecha de V está libre y al ser ejecutada desplazará a V hacia la derecha tantos cuadros como sea posible, sin pasar por sobre otro vehículo o salirse de la cuadrícula. La acción $left(V)$ se define en forma análoga a $left(V)$, y asimismo las acciones $up(V)$ y $down(V)$, que están disponibles solo para los vehículos dispuestos en posición vertical.

3. Parte 1: Rush Hour en el Cálculo de Situaciones

En la primera parte, usted deberá completar el programa `rush_hour.pl` disponible en la página del curso de tal forma de que las siguientes consultas funcionen respetando la dinámica del juego. Este archivo está fuertemente basado en el ejemplo `blocks.pl` que se vio en clases.

1. `holds(F,S)`: Cuando `S` es un término de situación instanciado y `F` es una variable (libre), `F` se debe instanciar con un término que se cumple en `S`. `F` es de la forma `at(V,X,Y)` o de la forma `blocked(X,Y)`, donde el primer hecho indica que la "cabeza" del vehículo `V` está en (`X`, `Y`), mientras que el segundo indica que (`X`, `Y`) no está disponible. Si usted desea, puede agregar más fluentes, pero al menos los mencionados arriba deben aparecer al realizar esta consulta.

2. `poss(A,S)`: Cuando `S` es un término de situación instanciado y `A` es una variable, `A` se instancia con una acción que es posible en `S`.

Para ambas consultas, al resatisfacer con ";", se espera Prolog entregue todas las posibilidades de instanciación, sin entrar en un loop infinito.

Observe que después de haber implementado ambas consultas correctamente, puede pedir a Prolog que resuelva cualquier puzle, haciendo la consulta `legal(S), holds(at(red,5,3),S)`.

4. Parte 2: Heurísticas para A*

Use la implementación de A* disponible desde el sitio web. Programe dos heurísticas eficientes, admisibles y no nulas. Evalúelas sobre 7 problemas de RHE a su elección y reporte el número de expansiones en un gráfico de barras.

5. Qué Entregar y Cómo

Envíe un email a `iic2613@ing.puc.cl` con tres archivos adjuntos. El primero, llamado `parte1.pl`, debe ser tal que se pueda evaluar que usted ha logrado completar la parte 1 de esta tarea. El segundo, llamado `parte2.pl`, que sea tal que al hacer la consulta `problema(N)` (donde `N` es un entero entre 1 y 7) ejecute A* sobre el problema número `N` definido por usted con ambas heurísticas. Finalmente, el tercer archivo deberá contener su informe para la parte 2 con una explicación detallada de sus heurísticas, una justificación de por qué son admisibles y el gráfico. No incluya el archivo `astar.pl`.