



# UNIVERSIDAD DE GRANADA

TÉCNICAS DE LOS SISTEMAS INTELIGENTES  
GRADO EN INGENIERÍA INFORMÁTICA

---

## PRÁCTICA 3

### PLANIFICACIÓN CLÁSICA (PDDL)

---

#### **Autor**

José María Sánchez Guerrero

#### **Rama**

Computación y Sistemas Inteligentes || Grupo 2 - Pablo Mesejo



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

CURSO 2019-2020

## Índice

Ejercicio 1	1
Ejercicio 2	2
Ejercicio 3	2
Ejercicio 4	2
Ejercicio 5	2
Ejercicio 6	2

## Introducción

En esta práctica tendremos que realizar varias tareas en el mundo de *StarCraft*, confeccionando un dominio de planificación clásico mediante el lenguaje PDDL. Estos ficheros se ejecutarán en el planificador *Metric-FF* que nos sacará un plan válido.

Se dispondrá de un par de ficheros por cada ejercicio (dominio y problema), los cuales ejecutaremos con la siguiente orden:

```
./ff -o <ruta-dominio>-f <ruta-problema>-s 0
```

El código para cada uno de los ejercicios es incremental con respecto al anterior, es decir, voy metiendo pequeñas modificaciones dependiendo de lo que se nos pida. Esto no sucede en el último ejercicio, ya que he tenido que cambiar varias cosas para que se ejecute en un tiempo razonable. Los cambios los explicaré más detalladamente después.

## Ejercicio 1

En este ejercicio se nos pide que implementemos todo el dominio de *StarCraft*, es decir, edificios, unidades, recursos y acciones básicas como moverse, asignar o construir edificios.

La declaración de los tipos, las constantes y predicados no tiene mucho misterio, así que viendo el código puede verse fácilmente cómo y por qué están declaradas. En cuanto a las acciones si las vamos a explicar mejor:

- **Navegar.** Mueve una unidad entre una localización y otra. Para ello, primero comprueba si la localización en la que se encuentra está conectada con la localización a la que va; y después se asegura que la unidad no estaba extrayendo ningún recurso. Si esto se cumple, añade la unidad a la nueva localización y la elimina de la anterior
- **Asignar.** Pone a un VCE a extraer recursos de un nodo. Si una unidad del tipo VCE se encuentra en la posición del recurso a extraer y no lo estaba extrayendo previamente, comienza a extraerlo.
- **Construir.** Ordena a un trabajador libre que construya un edificio. Al igual que antes, este trabajador tiene que ser un VCE, estar en la localización donde se va a construir y no puede estar ocupado extrayendo. Además, tenemos

que comprobar que no hay ningún edificio ya construido en esa localización, y para ello hemos utilizado el predicado '**exists**', que busca en el resto de localizaciones ese edificio. Por último, también tiene que existir otra unidad distinta, extrayendo el recurso que necesita el edificio para poder construirse.

Por otro lado, tenemos el fichero problema, en el cual declaramos y colocamos en el mapa tanto las unidades, como los edificios, como los minerales. Como el mundo está representado en forma de cuadrícula, también tendremos que definirla en el problema (en mi caso la hice gracias a la herramienta del editor online).

## **Ejercicio 2**

## **Ejercicio 3**

## **Ejercicio 4**

## **Ejercicio 5**

## **Ejercicio 6**