FastForward

- Fast Forward es un planificador que permite ejecutar planes definidos en el lenguaje PDDL
- El programa se puede descargar de http://fai.cs.uni-saarland.de/hoffmann/ff.html , hay un ejecutable para windows, para linux tendréis que compilar los fuentes.
- La versión Metric-Fast Forward que necesitaréis para los apartados finales de la práctica la podéis descargar de http://fai.cs.uni-saarland.de/hoffmann/metric-ff.html, esta versión la tendréis que compilar tanto para linux como para windows (necesitaréis tener instalados flex y bison)

Dominio + Problema

- Un problema de planificación en PDDL se divide en dos conjuntos de definiciones:
  1. Definición del dominio: Operadores, predicados, tipos, ...
  2. Definición de una instancia del problema: Constantes, hechos y objetivo
- Para ejecutar un plan en FF cada conjunto de definiciones está en un fichero separado.

Fichero de dominio (Definiciones)

- Requerimientos: Características del lenguaje PDDL que se usarán en la definición del problema
  - `:strips`, lenguaje básico
  - `:typing`, definición de tipos en parámetros y constantes
  - `:adl`, lenguaje avanzado con cuantificadores, efectos condicionales, ...
  - `:equality`, uso de comparaciones de igualdad
  - `:fluents`, lenguaje avanzado con funciones de coste y optimización
- Tipos: Tipos de las constantes/parámetros que se usarán en el problema
- Predicados: Predicados que representan el problema
- Acciones: Operadores de planificación para resolver el problema

Sintaxis del Dominio

```
(define (domain nom-domain)

 (:requirements :adl :typing)

 (:types  tipo1 tipo2)

 (:predicates
       (pred1 ?V1 - tipo1)
       (pred2 ?V1 - tipo2)
 )

 (:action nom-action1
        ...
 )

 (:action nom-action2
        ...
 )
)
```

Sintaxis de las acciones

```
(:action nom-action1
   :parameters (?V1 - tipo1 ?V2 - tipo2)
   :precondition (and
                    (pred1 ?V1)
                    (exists (?V3 - tipo2) (pred2 ?V3))))
                 )
   :effect (not (pred2 ?V2))
)
```

Definición de las acciones

- En la precondición
  - Un predicado atómico (strips)
  - Conjunción (`and`) de predicados atómicos (strips)
  - Combinación de condiciones con operadores lógicos (`and`,`or`,`not`) (adl)
  - Fórmulas cuantificadas (`forall`,`exists`) (adl)
- En los efectos
  - Un predicado atómico (strips)
  - La negación de un predicado atómico (strips)
  - La conjunción de predicados atómicos (strips)
  - Efectos condicionales (`when`) (adl)
  - Efectos cuantificados universalmente (`forall`) (adl)

Fichero del problema (Definiciones)

- `:domain`, dominio del problema
- `:objects`, constantes del problema
- `:init`, estado inicial del problema
- `:goal`, objetivo del problema

Sintaxis del problema

```
(define (problem nom-problem)
 (:domain nom-domain)
 (:objects obj1T1 obj2T1 obj3T1 - tipo1
           obj1T2 obj2T2 obj3T2 - tipo2
 )
 (:init
        (pred1 obj1T1)
        (pred1 obj2T1)
        (pred1 obj3T1)
        (pred2 obj1T2)
        (pred2 obj2T2)
        (pred2 obj3T2)
 )
 (:goal (forall (?V - tipo1)
                (pred1 ?V)
        )
 )
)
```

Evaluación de Condiciones

- Para ganar eficiencia es importante el orden en el que se escriben las condiciones
- Esto hay que tenerlo en cuenta en las condiciones de los operadores y en el objetivo
- Lo habitual es poner las condiciones mas restrictivas primero
- También es necesario evitar formulas que generen unificaciones demasiado complejas, sobre todo en la conclusion (por ejemplo con muchas disyunciones o existenciales)

Fluentes

- La extensión `:fluents` permite extender el lenguaje para incluir:
  - Funciones (de hecho son variables numéricas)
  - Comparaciones aritméticas entre funciones
  - Operaciones aritméticas entre funciones
  - Añadir costes en los efectos de las acciones
  - Optimización del plan a partir de los valores de las funciones

Fluentes en el fichero de dominio

- Se ha de declarar en el apartado de requerimientos `:fluents`
- Se ha de especificar una sección de funciones

```
1    (:functions
2     (funcion1 ?V - tipo1)
3     (funcion2 ?V - tipo2)
4     (coste-total)
5    )
```

- Los valores de las funciones se pueden consultar y comparar en las precondiciones de las acciones

```
(>= (+ (funcion1 ?V1) (funcion2 ?V1)) 10)
```

- Se puede modificar el valor en los efectos de las acciones

```
(:effect (increase (coste-total) 1))
```

Fluentes en el fichero del problema

- Se han de inicializar los valores de los fluentes (para cada constante si es necesario)

```
(:init
    ...
    (= (funcion1 obj1T1) 0)
    (= (coste-total) 0)
    ...
```

- Hay que incluir una sección que indica qué fluente optimizar

```
(:metric minimize (coste-total))
```

- Se pueden optimizar múltiples fluentes usando una combinación linear de sus valores

```
(:metric minimize
        (+ (coste1) (+ (* 5 (coste2)) (* 3 (coste3)))))
```

Ejecución del Fast Forward

- La versión normal del Fast Forward
  `ff -o dominio.pddl -f problema.pddl`
- La versión con fluentes de Fast Forward tiene un flag `-O` que permite optimizar el fluente definido en el problema
  `ff -O -o dominio.pddl -f problema.pddl`
  Con la versión con fluentes se pueden ejecutar todos los problemas
- El algoritmo de búsqueda de Fast Forward para los fluentes se basa en una versión de best first, así que no encuentra la solución óptima para el fluente que se use