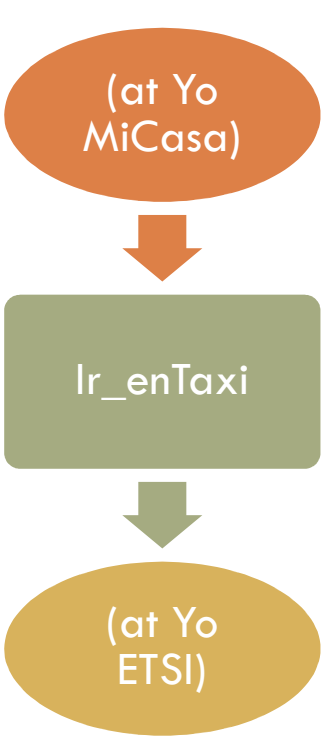
TÉCNICAS DE LOS SISTEMAS INTELIGENTES
PRÁCTICA3: PLANIFICACIÓN.

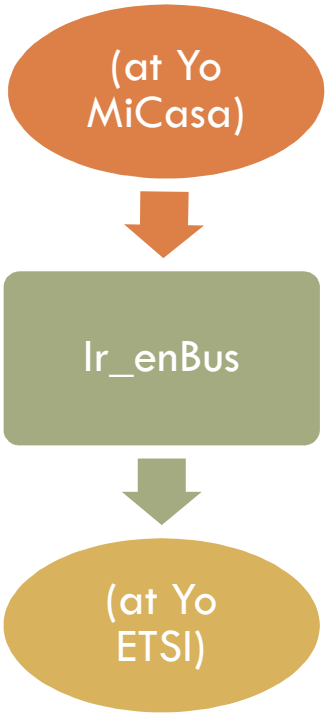
Sesión 2: Planificación HTN (Hierarchical Task Networks)
Lenguaje HTN-PDDL y Planificador IactivePlanner

Indice

- Conceptos básicos HTN
- Descripción de dominios y problemas HTN con HTN-PDDL
- Proceso del Planificador Iactive Planner
- Problemas propuestos.

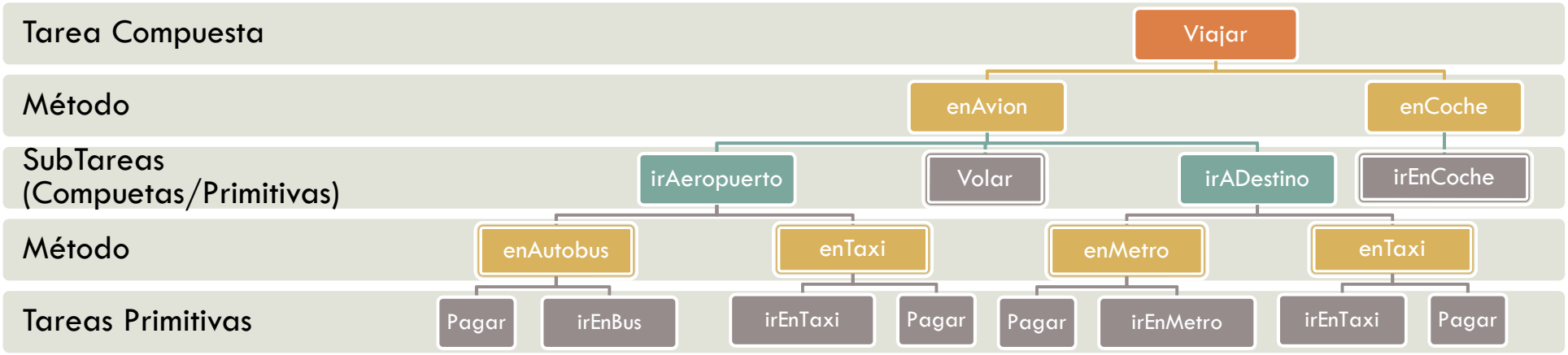
Planificación HTN (Hierarchical Task Networks)





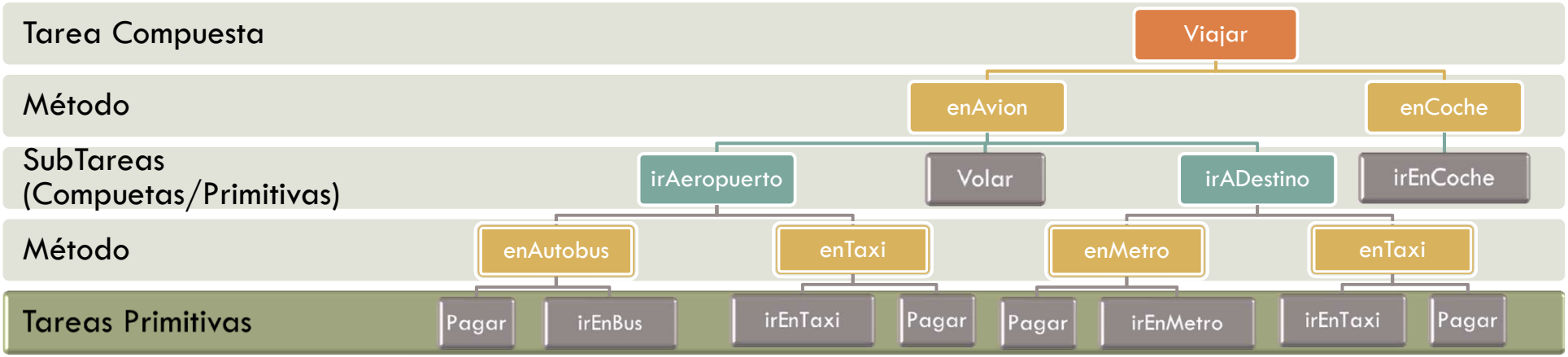
- La planificación automática basada en operadores (como en PDDL) se basa mucho en el análisis de relaciones causa/efecto
- Hay muchos problemas en los que sólo la representación y razonamiento basados en precondiciones/efectos no es suficiente.
- La **abstracción de tareas** permite representar conocimiento para que un planificador pueda usar otros medios para escoger qué acción aplicar

Planificación HTN: Conceptos



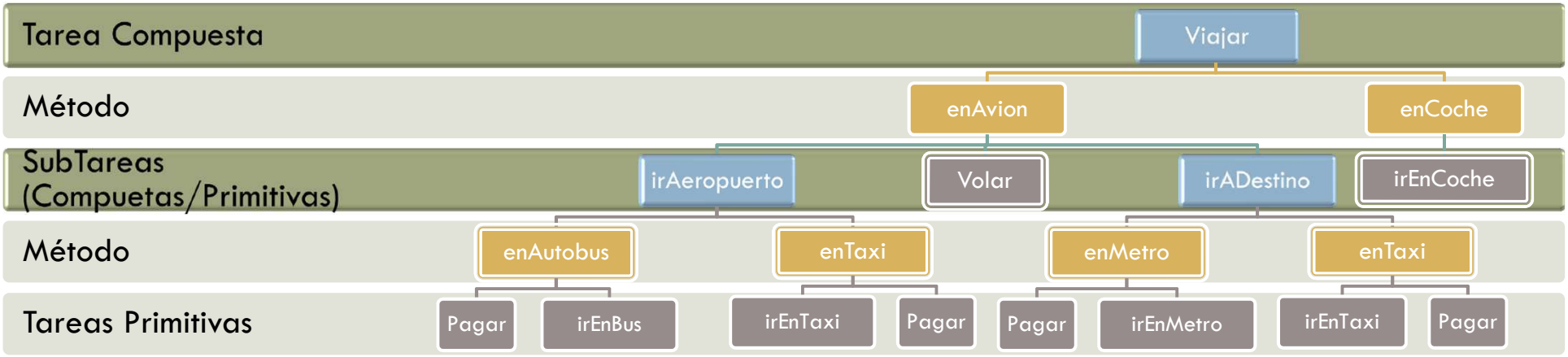
- **Dominio de planificación**
  - basado en representación de acciones a distintos niveles de abstracción
- Las acciones en cualquier nivel de abstracción se denominan **Tareas** (Tasks)
- Distinción entre tareas (acciones) **primitivas** y tareas **compuestas.**

Planificación HTN: Conceptos



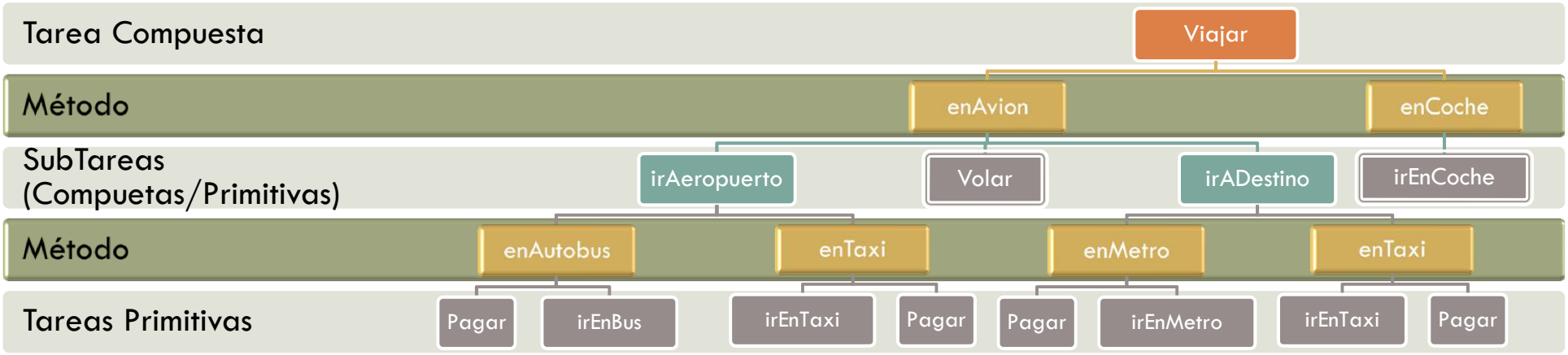
- Tarea Primitiva (en gris):
  - representa una acción del nivel de abstracción inferior, indivisible y cuya ejecución produce un cambio en el estado del mundo
  - Estado del mundo: conjunto de hechos (predicados instanciados)

Planificación HTN: Conceptos



- Tarea Compuesta (en azul):
  - Representa una acción o proceso de alto nivel que debe llevarse a cabo con la intervención de varias tareas de nivel inferior y que, normalmente, presenta distintas alternativas para su realización.

Planificación HTN: Conceptos



- Método de descomposición
  - Una alternativa o modo de llevar a cabo una tarea, representado como un conjunto de subtareas (compuestas o primitivas) y relaciones de orden entre ellas.
  - Describe qué pasos hay que seguir para descomponer una tarea en una secuencia de tareas primitivas
  - Una tarea compuesta tiene asociados varios métodos de descomposición: formas distintas de descomponer la tarea (por ejemplo *irAeropuerto*).

HTN-PDDL: Lenguaje para la representación de un dominio de planificación HTN

- HTN-PDDL es una extensión de PDDL para representar dominios de planificación HTN basados en tareas primitivas y compuestas

- estándar para representar acciones de dominios de planificación

PDDL

- lenguaje desarrollado por el Grupo de Sistemas Inteligentes del Departamento de Ciencias de la Computación e I.A. (DECSAI)

HTN-PDDL

- Es el lenguaje usado por el planificador IActivePlanner, también desarrollado por el grupo ISG
- Trasferido a la spin-off IActive Intelligent Technologies

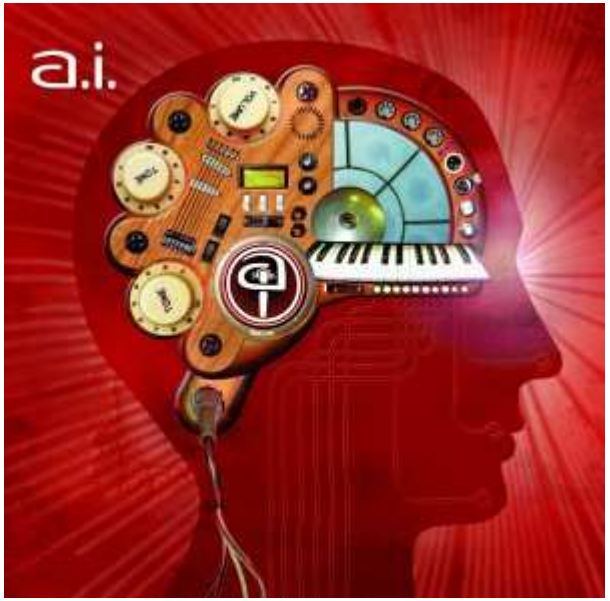
IActivePlanner

¿Tiene la IA cabida en el mercado?

ugr Universidad de Granada

Transferencia Tecnológica

IActive intelligent technologies



¿Oportunidades de Trabajo?

¿Desarrollo de carrera profesional?

¿Mercado?

Acciones primitivas: PDDL durative-actions

Conceptos

- Objetos del dominio:
  - constantes,
  - tipos
  - predicados,
  - funciones
- Acciones primitivas:
  - Representación PDDL
  - Parámetros con tipo,
  - precondiciones,
  - efectos,
  - duración

Código

```
(define (domain viajes)
  (:requirements  :typing  :fluents    :derived-predicates
     :negative-preconditions  :htn-expansion)
(:constants <….>)
(:types Persona Sitio - object
        Hogar Aeropuerto - Sitio
(:predicates (en ?p -  Persona ?s - Sitio))
(:functions (distancia ?x ?y - Sitio)
                (dinero ?p - Persona)
            (velocidad-taxi)
                (precio_km)
)
```

```
(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                                        (velocidad_taxi) )
:condition (and (en ?u ?o))
:effect (and (not (en ?u ?o))
                (en ?u ?d)))
```

Otros aspectos útiles de PDDL

Expresiones aritméticas

- Valores numéricos/funciones
- Especificar cómo calcular duraciones de acciones
- Condiciones con expresiones aritméticas

```
(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) )
 :condition (and (en ?u ?o)
                 (> (dinero ?u)
                    (* (precio_km)(distancia ?o ?d))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))
```

Otros aspectos útiles de PDDL

Predicados derivados

- Valores numéricos/funciones
- Derived literals: reglas de inferencia para "derivar" predicados de las precondiciones

```
(:derived (tiene_dinero ?p - Persona ?org ?dst - Sitio)
          ((> (dinero ?p)
              (* (precio_km)(distancia ?org ?dst)))
```

```
(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) )
 :condition (and (en ?u ?o)
                 (tiene_dinero ?u ?o ?d))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))
```

Otros aspectos útiles de PDDL

Operaciones sobre funciones.

- Valores numéricos/funciones
- Derived literals: reglas de inferencia para "derivar" predicados de las precondiciones
- Asignación, incremento/decremento de funciones

```
(:derived (tiene_dinero ?p - Persona ?org ?dst - Sitio)
            ((> (dinero ?u)
                        (* (precio_km)(distancia ?o ?d)))

(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                           (velocidad_taxi) )
 :condition (and (en ?u ?o)
                  (tiene_dinero ?u ?o ?d))
 :effect (and (not (en ?u ?o))
               (en ?u ?d)))

(:durative-action pagar
 :parameters (?u - Persona ?c - number)
 :duration (= ?dur 1)
 :condition (> (- (dinero ?u) ?c) 0)
 :effect (decrease (dinero ?u) ?c)
```

Tareas Compuestas

- Tarea compuesta
- Varios métodos
  - Precondición
  - Descomposición
- Tareas "inline"
  - Crear acciones "al vuelo" para un uso muy específico
  - Inferir nuevo conocimiento y añadirlo al estado el mundo
- Relaciones de orden
  - (<t1> <t2>)
  - [<t1> <t2>]

```
(:task irAeropuerto
 :parameters (?p - Persona ?c - Hogar ?a - Aeropuerto)
 (:method enTaxi
  :precondition ()
  :tasks (
              (ir_en_taxi ?p ?c ?a)
              (:inline (bind ?tarifa (* (distancia ?c ?a)
                                        (precio_km))) ())
              (pagar ?p ?tarifa)
             )
  (:method enBus
   :precondition ()
   :tasks ((:inline (bind ?tarifa (tarifa-bus)) () )
            (pagar ?p ?tarifa)
            (ir_en_bus ?p ?c ?a))
  (:method Andando
   :precondition ()
   :tasks (irAndando ?p ?c ?a))
);;task
```

Tareas Compuestas

```
(:task irAeropuerto
 :parameters (?p - Persona ?c - Hogar ?a - Aeropuerto)
 (:method enTaxi
  :precondition ()
  :tasks (
              (ir_en_taxi ?p ?c ?a)
              (:inline (bind ?tarifa (* (distancia ?c ?a)
                                        (precio_km))) ())
              (pagar ?p ?tarifa)
             )
  (:method enBus
   :precondition ()
   :tasks ((:inline (bind ?tarifa (tarifa-bus)) () )
           (pagar ?p ?tarifa)
           (ir_en_bus ?p ?c ?a))
  (:method Andando
   :precondition ()
   :tasks (irAndando ?p ?c ?a))
);;task
```

```
(:task Viajar
 :parameters (?p - Persona
              ?x ?y - Sitio)
 (:method enAvion
  :precondition (tiene prisa ?p)
  :tasks (
           (irAeropuerto ?p ?x GarciaLorca)
           (Volar ?p GarciaLorca Barajas)
           (irDestino ?p Barajas ?y)
          )
 (:method enCoche
  :precondition (not (tiene_prisa ?p))
  :tasks (irEnCoche ?p ?x ?y))
)
```

Problema HTN:

- se describe a partir de un estado inicial y de un objetivo representado como una tarea de alto nivel a llevar a cabo.

```
(define (problem UnViaje) (:domain Viajes)


(:objects
   MiCasa CasaMiPrimo - Hogar
   GarciaLorca Barajas - Aeropuerto
   Yo - Persona
)
(:init
 (en Yo MiCasa)
 (= (dinero Yo) 100)
 (= (distancia MiCasa GarciaLorca) 20)
 (= (precio-km) 7)
)
(:tasks-goal
   :tasks( (Viajar Yo MiCasa CasaMiPrimo))

  )
)
```

Esquema básico

Modelo del mundo

Conocimiento representado en HTN-PDDL

- Dominio
- Problema

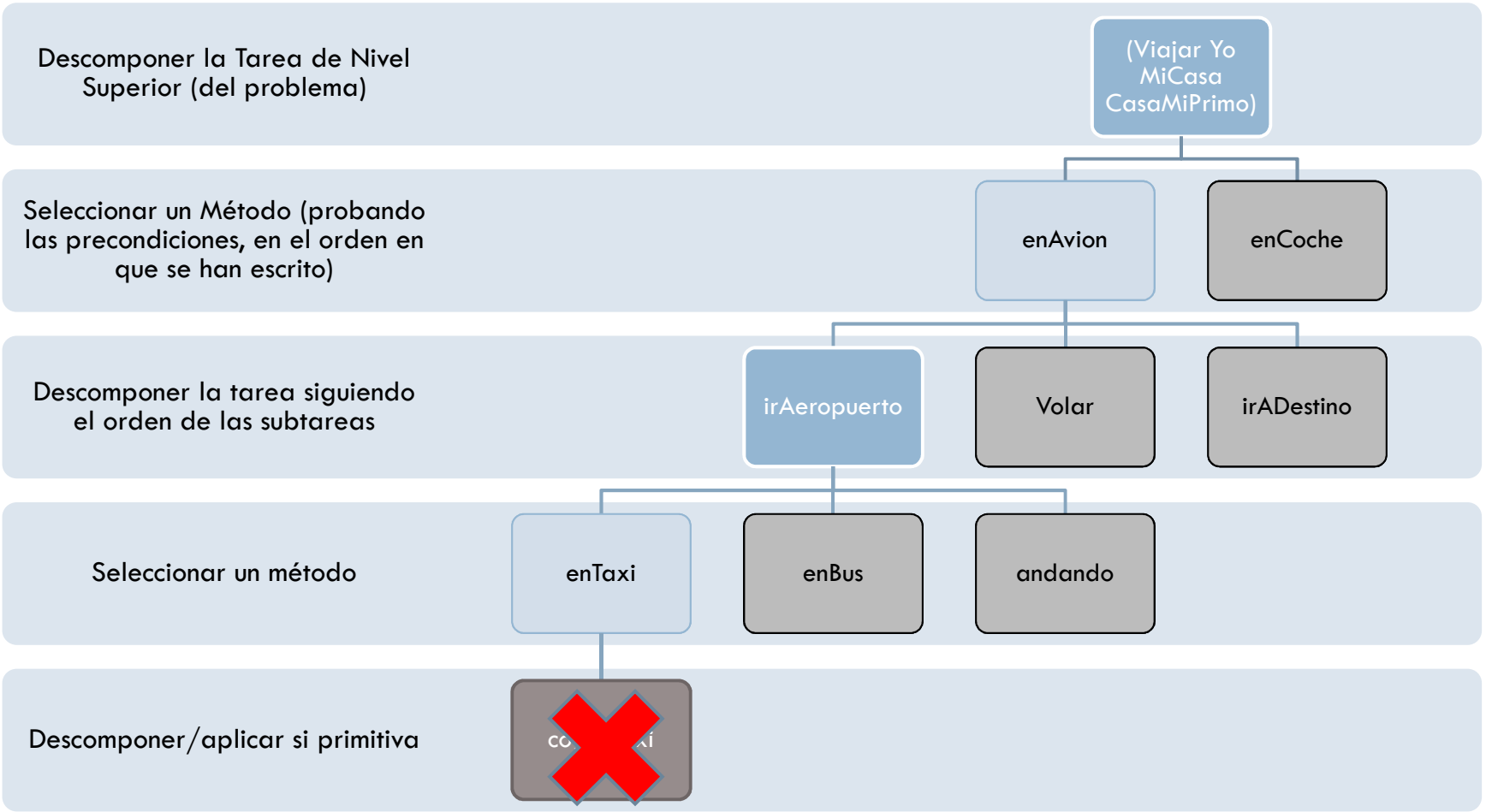
IActivePlanner

- Búsqueda
- Razonamiento

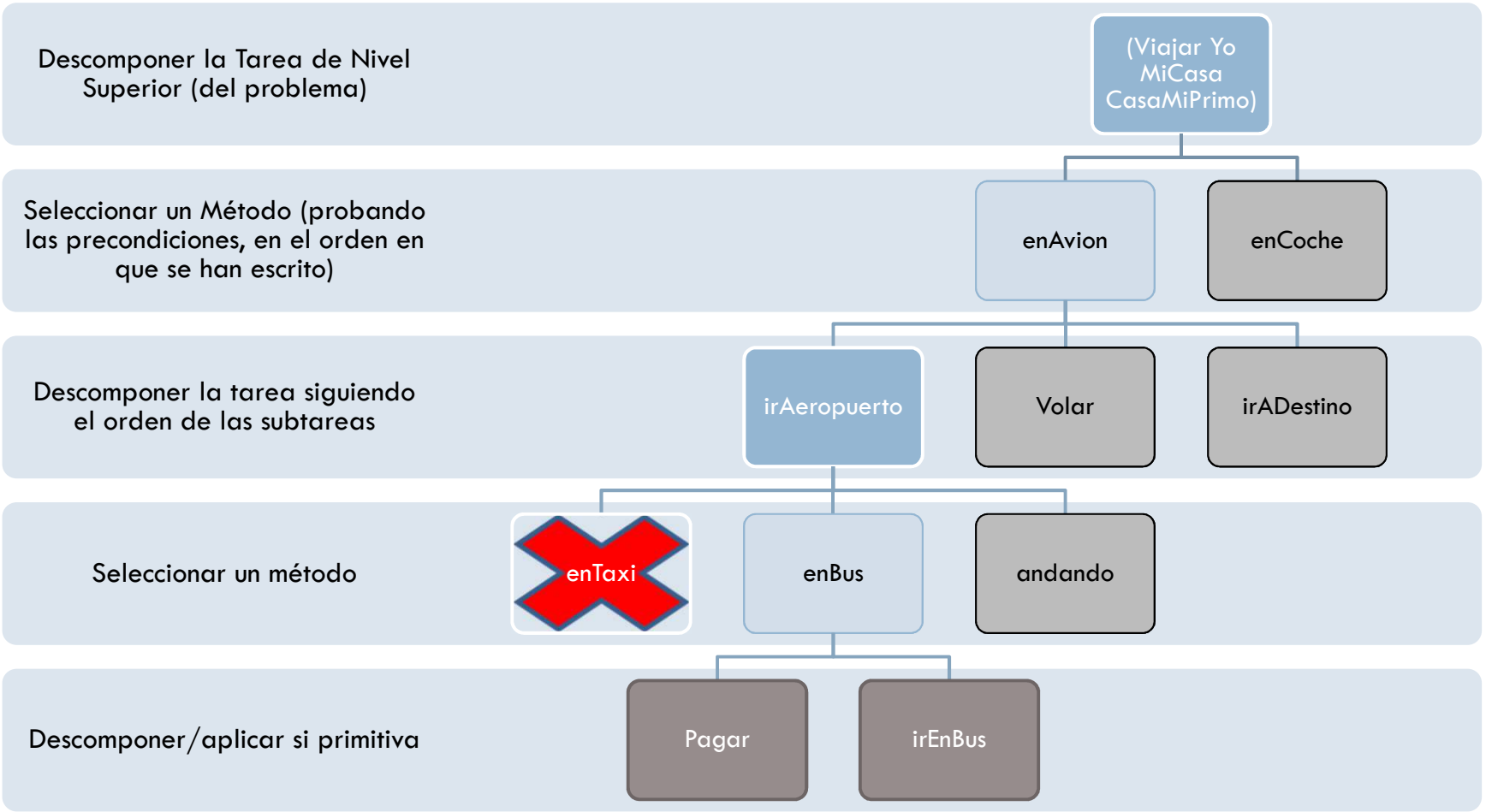
Plan

- Secuencia de acciones

Proceso de planificación HTN: Planificador IActivePlanner



Proceso de planificación HTN: Planificador IactivePlanner



Para qué sirven estas técnicas

- No adecuadas para resolver problemas combinatorios
  - Asignación horaria de vuelos en un aeropuerto
    - Explosión combinatoria
    - Mejor técnicas como CSPs.
- Adecuadas para problemas
  - Know-how pre-existente
    - Conocimiento experto sobre cómo realizar una tarea.
  - Adoptar estrategias similares a los humanos
    - Emergencias, Militar, Sanidad, VideoJuegos…

Para qué sirven estas técnicas

CONOCIMIENTO MÉDICO

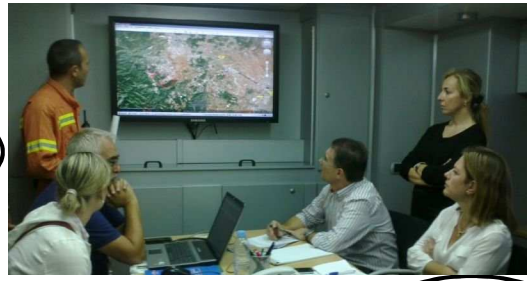
Perfil de paciente → Especialista Médico → Decide → TRATAMIENTO MÉDICO

CURAR AL PACIENTE → Especialista Médico

CONOCIMIENTO DOCENTE

Perfil de Alumno → PROFESOR → Decide → ITINERARIO FORMATIVO

ADQUIRIR CONTENIDOS → PROFESOR



CONOCIMIENTO SOBRE INCENDIOS

Situación Incendio → TÉCNICO EXTINCIÓN → Decide → PLAN DE EXTINCIÓN

APAGAR INCENDIO → TÉCNICO EXTINCIÓN

CONOCIMIENTO CIUDAD

Perfil Turista → ASESOR TURÍSTICO → Decide → PLAN DE VISITAS

VISITAR CIUDAD → ASESOR TURÍSTICO

IActivePlanner y la Transferencia Tecnológica

- Planes para extinguir incendios

IActive intelligent technologies

SIADEX

- E-Learning
- Generación Cursos para distintos perfiles de alumnos

ADAPTAPLAN

- Planes de tratamiento en oncología

IActive intelligent technologies

OncoTheraper Cognocare

- Planes de turismo personalizados

IActive intelligent technologies

SMARTOURISM Nativoo

- Planificación multiagente para múltiples patologías.

PlanInteraction

- Generación Automática de Música
- Juegos de estrategia (Unreal Tournament)
- Control de robots alto nivel.

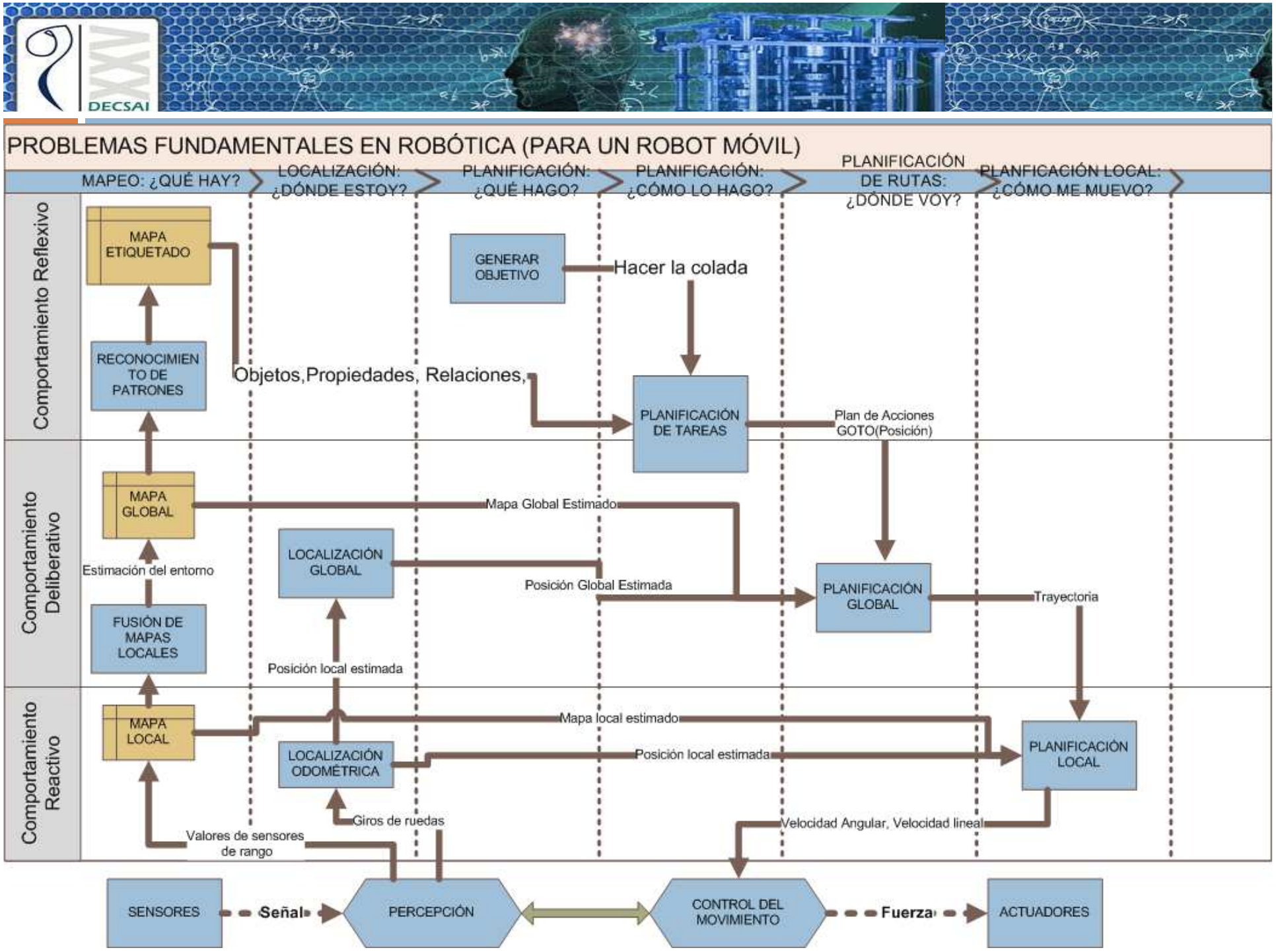
Proyectos Fin de Carrera

- Non-player characters (NPCs):
  - Move in game-world space (2D/3D)
  - Act in game-world
    - Pick-up objects, attack, hide, etc.
  - Exhibit behavior
    - Follow the player, get scared and runaway, etc.
- How do they think?
  - Navigation: Pathfinding
  - Action-driven behavior:
    Finite State Machines / Behavior Trees / Goal Oriented Action Planning / Utility systems

Planificación Clásica

- Overview on aigamedev.com:planning-in-games

- Overview on aigamedev.com:planning-in-games

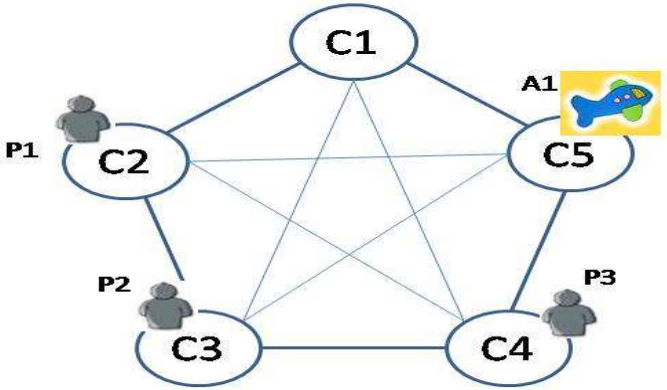

PROBLEMAS FUNDAMENTALES EN ROBÓTICA (PARA UN ROBOT MÓVIL)
MAPEO: ¿QUÉ HAY?
LOCALIZACIÓN: ¿DÓNDE ESTOY?
PLANIFICACIÓN: ¿QUÉ HAGO?
PLANIFICACIÓN: ¿CÓMO LO HAGO?
PLANIFICACIÓN DE RUTAS: ¿DÓNDE VOY?
PLANFICACIÓN LOCAL: ¿CÓMO ME MUEVO?
Comportamiento Reflexivo
Comportamiento Deliberativo
Comportamiento Reactivo
MAPA ETIQUETADO
RECONOCIMIENTO DE PATRONES
GENERAR OBJETIVO
Hacer la colada
Objetos,Propiedades, Relaciones,
PLANIFICACIÓN DE TAREAS
Plan de Acciones GOTO(Posición)
MAPA GLOBAL
Mapa Global Estimado
LOCALIZACIÓN GLOBAL
Estimación del entorno
Posición Global Estimada
PLANIFICACIÓN GLOBAL
Trayectoria
FUSIÓN DE MAPAS LOCALES
Posición local estimada
MAPA LOCAL
Mapa local estimado
LOCALIZACIÓN ODOMÉTRICA
Posición local estimada
PLANIFICACIÓN LOCAL
Giros de ruedas
Valores de sensores de rango
Velocidad Angular, Velocidad lineal
SENSORES
Señal
PERCEPCIÓN
CONTROL DEL MOVIMIENTO
Fuerza
ACTUADORES

Ejercicios

- Definir un dominio HTN de forma incremental
  - No vamos a partir de cero para escribir este dominio.
  - Partimos de un conjunto de tareas primitivas (acciones PDDL) ya conocidas.
  - Tareas compuestas incompletas
  - El dominio final tiene que resolver tres problemas
  - Resolver problemas en el dominio ZenoTravel (estándar para contrastar planificadores)

Dominio ZenoTravel

- Transporte aéreo entre ciudades
- Personas, ciudades, aviones
- En concreto: 5 ciudades, 1 avión, 3 personas



Acciones primitivas

- **Embarcar** una *persona* en un *avión* en una *ciudad* concreta.
- **Desembarcar** una *persona* en un *avión* en una *ciudad* concreta.
- **Volar** un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad lenta*
- **Volar** un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad rápida*
- **Repostar** un *avión* en una *ciudad*.

Preámbulo, tipos, predicados y funciones

```
(define (domain zeno-travel)
(:requirements
  :typing
  :fluents
 :derived-predicates
 :negative-preconditions
 :universal-preconditions
 :disjuntive-preconditions
 :conditional-effects
 :htn-expansion
  ; Requisitos adicionales para el
   manejo del tiempo
 :durative-actions
 :metatags
)
```

- Este preámbulo debe respetarse tal cual.

```
(:types aircraft person city - object)
(:constants slow fast - object)
(:predicates
        (at ?x - (either person aircraft) ?c - city)
             (in ?p - person ?a - aircraft)
             (diferente ?x ?y) ) ;;predicado derivado, ver más abajo
             (igual ?x ?y) ;;predicado derivado, ver más abajo
             (hay-fuel ?a ?c1 ?c2) ;;predicado derivado, ver más abajo
             )
```

```
(:functions
        (fuel ?a - aircraft) ;;cantidad de fuel actual de un avión
        (distance ?c1 - city ?c2 - city) ;;distancia entre dos ciudades
        (slow-speed ?a - aircraft); ;;velocidad "lenta" de un avión
        (fast-speed ?a - aircraft)  ;;velocidad "rápida" de un avión
        (slow-burn ?a - aircraft);;razón de consumo de un avión a velocidad lenta
        (fast-burn ?a - aircraft);razón de consumo de un avión a velocidad rápida
        (capacity ?a - aircraft) ;;capacidad de fuel de un avión
        (refuel-rate ?a - aircraft) ;;razón de repostaje de un avión (para calcular
  ;; el tiempo de repostaje
        (total-fuel-used) ;;valor del fuel total usado
        (boarding-time) ;;valor constante de tiempo de embarque
        (debarking-time) ;;valor constante de tiempo de desembarque
        )
```


```
(:durative-action board
 :parameters (?p - person ?a - aircraft ?c - city)
 :duration (= ?duration (boarding-time))
 :condition (and  (at ?p ?c)
                  (at ?a ?c))
 :effect (and  (not (at ?p ?c))
               (in ?p ?a)))
```

Desembarcar una *persona* en un *avión* en una *ciudad* concreta.

```
(:durative-action debark
 :parameters (?p - person ?a - aircraft ?c - city)
 :duration (= ?duration (debarking-time))
 :condition (and (in ?p ?a)
                 (at ?a ?c))
 :effect (and  (not (in ?p ?a))
               (at ?p ?c)))
```

Volar un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad lenta*

```
(:durative-action fly
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :duration (= ?duration (/ (distance ?c1 ?c2) (slow-speed ?a)))
 :condition (and (at ?a ?c1)
                 (>= (fuel ?a)(* (distance ?c1 ?c2) (slow-burn ?a))))
 :effect (and  (not (at ?a ?c1))
               (at ?a ?c2)
               (increase (total-fuel-used)
                         (* (distance ?c1 ?c2) (slow-burn ?a)))
               (decrease (fuel ?a)
                         (* (distance ?c1 ?c2) (slow-burn ?a)))))
```

Volar un *avión* de una *ciudad origen* a una *ciudad destino* a una *velocidad rápida*

```
(:durative-action zoom
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :duration (= ?duration (/ (distance ?c1 ?c2) (fast-speed ?a)))
 :condition (and (at ?a ?c1)
                 (>= (fuel ?a) (* (distance ?c1 ?c2) (fast-burn ?a))))
 :effect (and (not (at ?a ?c1))
              (at ?a ?c2)
              (increase (total-fuel-used)
                        (* (distance ?c1 ?c2) (fast-burn ?a)))
              (decrease (fuel ?a)
                        (* (distance ?c1 ?c2) (fast-burn ?a)))))
```

Repostar un *avión* en una *ciudad*.

```
(:durative-action refuel
:parameters (?a - aircraft ?c - city)
:duration (= ?duration (/(- (capacity ?a) (fuel ?a))(refuel-rate ?a)))
:condition (and  (> (capacity ?a) (fuel ?a))
                 (at ?a ?c))
:effect (assign (fuel ?a) (capacity ?a)))
```

Predicados derivados

- *;; el consecuente "vacío" se representa como "()" y significa "siempre verdad"*
- *;;un objeto es siempre igual a sí mismo*

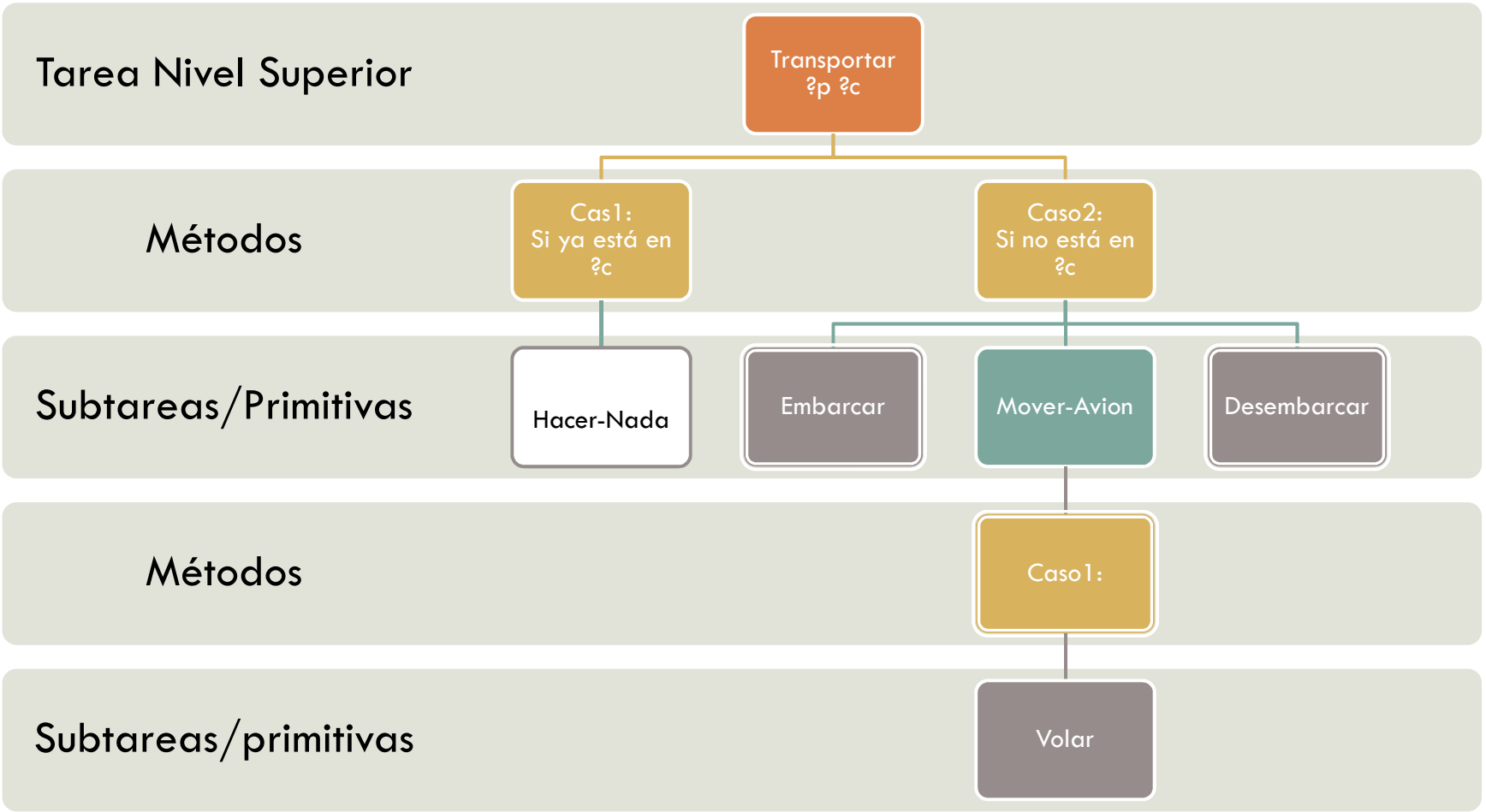
```
(:derived
   (igual ?x ?x) ())
```

- *;; dos objetos son diferentes si no son iguales*

```
(:derived
   (diferente ?x ?y) (not (igual ?x ?y)))

(:derived
    (hay-fuel ?a - aircraft ?c1 - city ?c2 - city)
    (> (fuel ?a) 1))
```

Tareas Compuestas



Transportar una persona a una ciudad destino

```
(:task transport-person
   :parameters (?p - person ?c - city)
   (:method Case1 ; si la persona esá en la ciudad no se hace nada
    :precondition (at ?p ?c)
    :tasks ()
   )
 ;si la persona no está en la ciudad destino, pero avion y persona están en la
   misma ciudad
  (:method Case2
    :precondition (and (at ?p - person ?c1 - city)
                       (at ?a - aircraft ?c1 - city))

    :tasks (
            (board ?p ?a ?c1)
            (mover-avion ?a ?c1 ?c)
            (debark ?p ?a ?c )))
   )
```

Mover avión

```
(:task mover-avion
:parameters (?a - aircraft ?c1 - city ?c2 -city)
(:method fuel-suficiente
  :precondition (hay-fuel ?a ?c1 ?c2)
  :tasks (
         (fly ?a ?c1 ?c2)
        )
   )
  )
```

- Comprobar que con este dominio básico se resuelve el problema siguiente:

```
(define (problem zeno-0)
(:domain zeno-travel)
(:customization
(= :time-format "%d/%m/%Y %H:%M:%S")
(= :time-horizon-relative 2500)
(= :time-start "05/06/2007 08:00:00")
(= :time-unit :hours))

(:objects
   p1 p2 p3 p4 - person
   c1 c2 c3 c4 c5 - city
   a1 - aircraft
)
(:init
   (at p1 c4)
   (at p2 c4)
   (at p3 c5)
   (at a1 c4)
```

```
(= (distance c1 c2) 100)
   (= (distance c2 c3) 100)
   (= (distance c3 c4) 100)
   (= (distance c4 c5) 100)
   (= (distance c5 c1) 100)
   (= (distance c1 c5) 100)

   (= (distance c1 c3) 150)
   (= (distance c1 c4) 150)
   (= (distance c2 c5) 150)
   (= (distance c2 c4) 150)
   (= (distance c3 c1) 150)
   (= (distance c3 c5) 150)
   (= (distance c4 c2) 150)
   (= (distance c4 c1) 150)
   (= (distance c5 c2) 150)
   (= (distance c5 c3) 150)
```

```
   (= (fuel a1) 100000)
   (= (slow-speed a1) 10)
   (= (fast-speed a1) 20)
   (= (slow-burn a1) 1)
   (= (fast-burn a1) 2)
   (= (capacity a1) 100000)
   (= (refuel-rate a1) 1)
   (= (total-fuel-used) 0)
   (= (boarding-time) 1)
   (= (debarking-time) 1)
 )
(:tasks-goal
   :tasks(
   (transport-person p1 c4)
   (transport-person p2 c5)
   (transport-person p3 c2))))
```

- PROBLEMA 1: Comprobar que NO se resuelve el problema siguiente y modificar el dominio

```
(define (problem zeno-0)
(:domain zeno-travel)
(:customization
(= :time-format "%d/%m/%Y %H:%M:%S")
(= :time-horizon-relative 2500)
(= :time-start "05/06/2007 08:00:00")
(= :time-unit :hours))

(:objects
   p1 p2 p3 p4 - person
   c1 c2 c3 c4 c5 - city
   a1 - aircraft
)
(:init
   (at p1 c4)
   (at p2 c4)
   (at p3 c5)
   (at a1 c4)
```

```
(= (distance c1 c2) 100)
   (= (distance c2 c3) 100)
   (= (distance c3 c4) 100)
   (= (distance c4 c5) 100)
   (= (distance c5 c1) 100)
   (= (distance c1 c5) 100)

   (= (distance c1 c3) 150)
   (= (distance c1 c4) 150)
   (= (distance c2 c5) 150)
   (= (distance c2 c4) 150)
   (= (distance c3 c1) 150)
   (= (distance c3 c5) 150)
   (= (distance c4 c2) 150)
   (= (distance c4 c1) 150)
   (= (distance c5 c2) 150)
   (= (distance c5 c3) 150)
```

```
(= (fuel a1) 100000)
   (= (slow-speed a1) 10)
   (= (fast-speed a1) 20)
   (= (slow-burn a1) 1)
   (= (fast-burn a1) 2)
   (= (capacity a1) 100000)
   (= (refuel-rate a1) 1)
   (= (total-fuel-used) 0)
   (= (boarding-time) 1)
   (= (debarking-time) 1)
)

(:tasks-goal
   :tasks(
   (transport-person p1 c5)
   (transport-person p2 c5)
   (transport-person p3 c5)
```

- PROBLEMA 2: Comprobar que NO se resuelve el problema siguiente y volver a modificar el dominio

```
(define (problem zeno-0)
(:domain zeno-travel)
(:customization
(= :time-format "%d/%m/%Y %H:%M:%S")
(= :time-horizon-relative 2500)
(= :time-start "05/06/2007 08:00:00")
(= :time-unit :hours))

(:objects
   p1 p2 p3 p4 - person
   c1 c2 c3 c4 c5 - city
   a1 - aircraft
)
(:init
   (at p1 c4)
   (at p2 c4)
   (at p3 c5)
   (at a1 c4)
```

```
(= (distance c1 c2) 100)
   (= (distance c2 c3) 100)
   (= (distance c3 c4) 100)
   (= (distance c4 c5) 100)
   (= (distance c5 c1) 100)
   (= (distance c1 c5) 100)

   (= (distance c1 c3) 150)
   (= (distance c1 c4) 150)
   (= (distance c2 c5) 150)
   (= (distance c2 c4) 150)
   (= (distance c3 c1) 150)
   (= (distance c3 c5) 150)
   (= (distance c4 c2) 150)
   (= (distance c4 c1) 150)
   (= (distance c5 c2) 150)
   (= (distance c5 c3) 150)
```

```
(= (fuel a1) 200)
   (= (slow-speed a1) 10)
   (= (fast-speed a1) 20)
   (= (slow-burn a1) 1)
   (= (fast-burn a1) 2)
   (= (capacity a1) 300)
   (= (refuel-rate a1) 1)
   (= (total-fuel-used) 0)
   (= (boarding-time) 1)
   (= (debarking-time) 1)
 )
(:tasks-goal
   :tasks(
   (transport-person p1 c5)
   (transport-person p2 c5)
   (transport-person p3 c5)
```

- PROBLEMA ·3: Comprobar que NO se resuelve el problema siguiente y modificar el dominio

```
(define (problem zeno-0)
(:domain zeno-travel)
(:customization
(= :time-format "%d/%m/%Y %H:%M:%S")
(= :time-horizon-relative 2500)
(= :time-start "05/06/2007 08:00:00")
(= :time-unit :hours))

(:objects
   p1 p2 p3 p4 - person
   c1 c2 c3 c4 c5 - city
   a1 - aircraft
)
(:init
   (at p1 c4)
   (at p2 c4)
   (at p3 c5)
   (at a1 c4)
```

```
(= (distance c1 c2) 100)
   (= (distance c2 c3) 100)
   (= (distance c3 c4) 100)
   (= (distance c4 c5) 100)
   (= (distance c5 c1) 100)
   (= (distance c1 c5) 100)

   (= (distance c1 c3) 150)
   (= (distance c1 c4) 150)
   (= (distance c2 c5) 150)
   (= (distance c2 c4) 150)
   (= (distance c3 c1) 150)
   (= (distance c3 c5) 150)
   (= (distance c4 c2) 150)
   (= (distance c4 c1) 150)
   (= (distance c5 c2) 150)
   (= (distance c5 c3) 150)
```

```
(= (fuel-limit) 1500)
(= (fuel a1) 200)
   (= (slow-speed a1) 10)
   (= (fast-speed a1) 20)
   (= (slow-burn a1) 1)
   (= (fast-burn a1) 2)
   (= (capacity a1) 300)
   (= (refuel-rate a1) 1)
   (= (total-fuel-used) 0)
   (= (boarding-time) 1)
   (= (debarking-time) 1)
 )
(:tasks-goal
   :tasks(
   (transport-person p1 c5)
   (transport-person p2 c5)
   (transport-person p3 c5)
```

(domain …..

(:task transport-person

:parameters (?p - person ?c - city)

……

;si la persona no está en la ciudad destino, pero avion y persona están en la misma ciudad

(:method Case2

:precondition (and (at ?p - person ?c1 - city)

(at ?a - aircraft ?c1 - city))

:tasks (

(board ?p ?a ?c1)

(mover-avion ?a ?c1 ?c)

(debark ?p ?a ?c)))

…..

…..

(:import "Primitivas-Zenotravel.pddl")

)