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

**Tarea 2 – El mundo de las grúas**
IIC2613 - Inteligencia Artificial
Segundo Semestre, 2018
**Entrega:** 23 de octubre, hasta las 23:59

# 1. Objetivo

El objetivo de esta tarea es que usted practique conceptos de programación en lógica y búsqueda, aplicado a la inteligencia artificial.

# 2. El *mundo de las grúas*

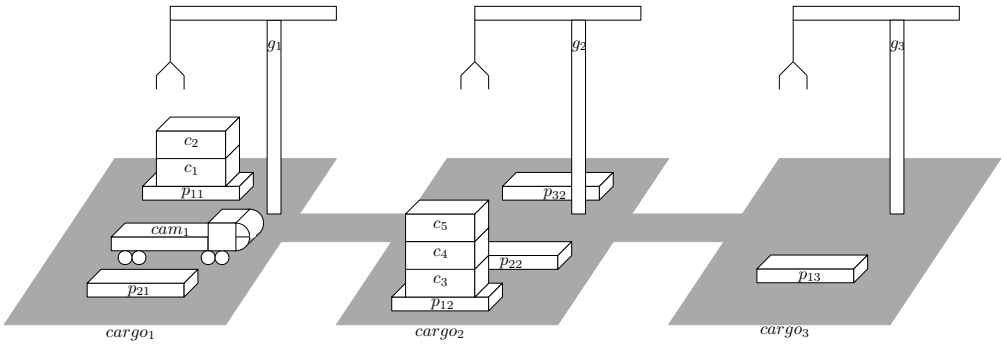
El mundo de las grúas es una simplificación de un puerto real. En este mundo existen *containers*, *paletas*, *grúas*, *áreas de carga* y *camiones*. Tal como en el mundo de bloques, los containers se pueden apilar unos sobre otros. A diferencia del mundo de bloques, el container inferior de una pila de containers se ubica sobre una paleta (no una mesa). En un área de carga siempre existe al menos una grúa y al menos una paleta. La grúa puede levantar un container a la vez y soltar un container que tiene levantado. La grúa también puede cargar un container que tiene levantado sobre un camión y puede descargar un container que se encuentra dentro de un camión. Para simplificar nuestro mundo, suponemos que el camión tiene capacidad infinita. Los camiones se mueven entre áreas de carga, ejecutando la acción manejar.

Para modelar este mundo, usamos los siguientes predicados dinámicos:

- $en(x, cargo)$, donde $x$ es una grúa, una paleta, un camión o un container, establece el hecho que $x$ está en el área de carga *cargo*. Si $x$ es un container $en(x, cargo)$ solo es verdadero cuando $x$ está sobre otro container o sobre una paleta (no cuando está levantado por la grúa ni cuando está cargado en un camión).
- $sobre(x, y)$, donde $x$ es un container e $y$ es un container o una paleta, establece que $x$ está directamente sobre $y$.
- $levantando(g, c)$, donde $g$ es una grúa y $c$ un container, establece el hecho que la grúa $g$ está levantando el container $c$. Recuerde que, según lo dicho arriba, una grúa solo puede levantar un container a la vez.
- $dentro(c, t)$, donde $c$ es un container y $t$ es un camión, establece que el container $c$ está dentro el camión $t$.
- $disponible(g)$, es un predicado opcional (úselo si es que desea hacerlo!), que es verdadero exactamente cuando la grúa $g$ no está levantando ningún container.
- $despejada(s)$, es un predicado opcional (úselo si es que desea hacerlo!), que es verdadero exactamente cuando el container/paleta $s$ no tienen un container directamente sobre él/ella.

Además, utilizamos las siguientes acciones:

- $levantar(gr, co, sup, lu)$: denota la acción en que la grúa levanta a un container $co$ que se encontraba directamente sobre $sup$ (que es un container o una paleta) en el lugar $lu$. Es posible si $gr$ y $co$ ambos están en $lu$, si $co$ se encuentra directamente sobre $sup$, y si $gr$ no está levantando ningún container. Como efecto, ahora $gr$ levanta a $co$, $co$ deja de estar sobre $sup$ y $co$ ya no está en el lugar $lu$ (si este último efecto suena extraño, lea nuevamente la definición del predicado $en$).
- $soltar(gr, co, sup, lu)$: denota la acción inversa de la anterior: la grua $gr$, que estaba levantando a $co$ lo deja directamente sobre $sup$. Los efectos y precondición se pueden obtener reflexionando sobre qué significa que sea la acción inversa a la anterior.
- $cargar(gr, co, ca, lu)$: denota la acción en que la grúa $gr$ carga el container $co$ en el camión $ca$ que está en el lugar $lu$. La acción solo es posible cuando $gr$ está levantando el container $co$ y cuando $ca$ y $gr$ ambas están en el lugar $lu$. Como resultado de ejecutarla, $gr$ ya no levanta a $co$ y $co$ está dentro de $ca$.
- $descargar(gr, co, ca, lu)$: denota la acción inversa de la anterior; es decir, la que descarga a $co$ que está dentro de $ca$. Los efectos y precondición se pueden obtener reflexionando sobre qué significa que sea la acción inversa a la anterior.




# 3. Parte 1: Modelando el mundo de las grúas

Complete el archivo base `tarea2_base.pl` disponible en el sitio del curso de manera que represente fielmente el mundo de las grúas, tal como fue descrito arriba. En este archivo se describe una situación inicial que modela al dibujo de la figura.

El predicado `legal` (definido en el archivo base) enumera situaciones alcanzables desde `s0`. De esta manera, una vez que usted complete esta parte, debiera poder hacer las siguientes consultas sobre la situación inicial definida en el archivo base (y que representa al dibujo del enunciado):[1]

```
?- poss(A,s0).
A = manejar(cam1, cargo1, cargo2) ;
A = levantar(g1, c2, c1, cargo1) ;
A = levantar(g2, c5, c4, cargo2) ;
false.

?- legal(S).
S = s0 ;
S = do(manejar(cam1, cargo1, cargo2), s0) ;
S = do(levantar(g1, c2, c1, cargo1), s0) ;
S = do(levantar(g2, c5, c4, cargo2), s0) ;
S = do(manejar(cam1, cargo2, cargo1), do(manejar(cam1, cargo1, cargo2), s0)) ;
```

[1] Tome en cuenta que las respuestas que usted obtenga pueden diferir, en orden, de lo que se muestra acá.

```
S = do(manejar(cam1, cargo2, cargo3), do(manejar(cam1, cargo1, cargo2), s0)) ;
S = do(levantar(g1, c2, c1, cargo1), do(manejar(cam1, cargo1, cargo2), s0)) ;
S = do(levantar(g2, c5, c4, cargo2), do(manejar(cam1, cargo1, cargo2), s0)) ;
S = do(manejar(cam1, cargo1, cargo2), do(levantar(g1, c2, c1, cargo1), s0)) ;
S = do(levantar(g2, c5, c4, cargo2), do(levantar(g1, c2, c1, cargo1), s0)) ;
S = do(soltar(g1, c2, p21, cargo1), do(levantar(g1, c2, c1, cargo1), s0)) ;
S = do(soltar(g1, c2, c1, cargo1), do(levantar(g1, c2, c1, cargo1), s0)) .
```

# 4. Parte 2: Usando A*

Como tal vez habrá notado, usar el predicado `legal(S)` para resolver problemas interesantes es completamente impráctico. Acá exploraremos cómo usar Weighted A* para resolver más rápidamente problemas interesantes. **Supondremos acá que los objetivos están definidos por una conjunción de hechos de la forma: "cierto producto $p$ debe ser entregado en $(x, y)$".**

Una implementación de A* está disponible en el sitio web: `astar.pl`. El objetivo de esta parte es que usted agregue un par de predicados que permitan usar A* sobre el mundo de bodegas.

Para que esta implementación funcione correctamente, usted deberá definir dos predicados:

- *goal_condition*$(G)$: define qué proposiciones deben ser verdaderas en un estado final
- *astar_heuristic*$(E, H)$: si $E$ es un conjunto de proposiciones (que definen un estado), $H$ unifica con el resultado de una heurística para este problema.

Por ejemplo, usando una heurística que el profesor definió (en su solución a esta tarea), y usando el objetivo definido por:

```
goal_condition([en(c1,cargo2),en(c5,cargo1)]).
```

se obtiene la siguiente respuesta:

```
?- astar(P), pretty_print_plan(P).
Plan length=12
A* expansions=354
levantar(g1,c2,c1,cargo1)
cargar(g1,c2,cam1,cargo1)
levantar(g1,c1,p11,cargo1)
cargar(g1,c1,cam1,cargo1)
levantar(g2,c5,c4,cargo2)
manejar(cam1,cargo1,cargo2)
cargar(g2,c5,cam1,cargo2)
descargar(g2,c1,cam1,cargo2)
manejar(cam1,cargo2,cargo1)
soltar(g2,c1,c4,cargo2)
descargar(g1,c5,cam1,cargo1)
soltar(g1,c5,p11,cargo1)
```

Espero que usted pueda encontrar una heurística que sea mejor que la mía!

# 5. Qué se entrega

Espero que usted entregue dos cosas:

1. El archivo `tarea2.pl` con el código de su tarea.
2. Un informe en donde:
   *a*) Describa la heurística que implementó para la parte 2, en donde argumente por qué es admisible.
   *b*) Un pequeño estudio experimental, que considere unos diez problemas definidos por usted, en donde reporte el número de estados expandidos por weighted A* usado con pesos 1, 1.5, 2 y 3. Acá usted puede modificar el mundo como quiera: agregar más grúas, más containers, más camiones y más áreas de carga. Describa en su informe el tipo de problemas que usó.

# 6. Bonus

Una deficiencia de la modelación que hemos usado es que no se consideran movimientos en paralelo de las grúas. Agregue a su dominio una acción que permita ejecutar carga y otra acción que permita ejecutar descarga paralela entre diferentes grúas. Una forma en que puede hacer esto es transformando los argumentos la acción levantar/soltar en una lista.