Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación
IIC2613-1 Inteligencia Artificial

Tarea 1

21 de 09 de 2022

Maximiliano Valdés

Parte I. DCCajotier

La solución que se encontró para el problema, asumiendo que los programadores del 1 al 5 están ordenados de izquierda a derecha, fue la siguiente:

$$combinacion(1,"Santiago","Mac","Clingo","Cafe","Jazz")$$

$$combinacion(2,"Antofagasta","Nokia","Javascript","RedBull","RockLatino")$$

$$combinacion(3,"Talca","Papel","C","Mate","CumbiaVillera")$$

$$combinacion(4,"Valparaiso","Tablet","Ruby","Gatorade","LoFi")$$

$$combinacion(5,"Concepcion","ENIAC","Python","Agua","DeAscensor")$$

La que se puede observar que cumple con todas las reglas planteadas.

Se descubrió que el programador 1 programa en clingo y que el programador 5 escucha música de ascensor.

Parte II. DCCursos

La lógica general del programa consiste en eliminar todos los casos no deseables por medio de restricciones, limitar el número de combinaciones según las variables correspondientes y forzar a que el número de cursos sin aprobar sea igual al número de cursos asignados a un semestre, para evitar soluciones donde queden cursos no aprobados sin asignar.

Respecto de la actividad uno, para que los cursos aprobados no pudiesen tener un semestre asignado, se implementaron tres restricciones que eliminasen todas las posibles combinaciones que incluyesen cursos aprobados, considerando cursos asignados a cuando se aprobaron, a un semestre distinto o semestres con cualquier otro curso aprobado.

Para hacer que todos los cursos no aprobados tuviesen un único semestre asignado, se usaron dos restricciones. La primera hace uso de la función #count de clingo, que permite contar cuantas ocurrencias hay de una cierta variable en una función, dándole el uso de

contar el total de cursos N3, los cursos aprobados N1 y los cursos a los que se les asignó un semestre N2, para poder eliminar el caso excluyente donde el número de cursos sin aprobar N3 - N1 es distinto del número de cursos con semestre asignado N2, forzando así a que sean iguales. La segunda regla consiste en que no se puede hacer más de una asignación por curso, por medio de la restricción de combinaciones 0{}1, lo que en conjunto con la primera regla, hace que todos los cursos no aprobados deban tener exactamente un semestre asignado.

Para que los semestres no pueden tener más de 5 ramos asignados se usa la restricción de combinaciones 0{}5.

Para que los cursos que sólo se dictan en semestres impares no pueden estar asignados a semestres pares y los cursos que sólo se dictan en semestres pares no pueden ser asignados a semestres impares simplemente se eliminan los casos en los que esto pasa, determinando si un semestre es par o impar con el operador resto de la división.

Ahora, respecto de la actividad dos, para que si un ramo A es prerrequisito de un ramo B, entonces el ramo A no puede ser asignado en un semestre igual o posterior al del ramo B, se deben eliminar los casos en el que el semestre en que se dicta A es mayor o igual al del ramo B. De igual forma para que si un ramo A es correquisito de un ramo B, entonces el ramo A no puede ser asignado en un semestre posterior al del ramo B, se deben eliminar los casos en los que el semestre asignado a A es mayor al asignado a B.

Parte III. DCCharcos

Charcos

El comportamiento general de los flotadores se basa en el código entregado del comportamiento de las cajas, con las mismas funciones salvo por las reglas que evitan que el robot pase por el agua sin flotador. No se modifican lineas del código preexistente, pero si se define la acción "$liftFloat$" junto con las otras acciones para no generar errores.

Primero se hace que los flotadores se muevan junto con el robot que los recoge, coordinando las funciones de posición del robot, flotador y de flotador en robot, desde el momento en que el robot recoge el flotador, por medio de combinatorias gatilladas por recoger el flotador, añadiendo que la posición del flotador debe ser igual a la del robot que lo recogió de ahí en adelante y que Z = 1 para ese flotador.

En segundo lugar se plantean las reglas necesarias para la acción de recoger el flotador. Para esto se plantean todas las combinaciones de posiciones adyacentes del flotador al robot, para T=T y T=0, para luego definir la función asociada a la acción de recoger el flotador en caso de que haya flotadores adyacentes.

Lo que sigue son una serie de restricciones que aseguran el correcto funcionamiento del código, incluyendo evitar recoger flotadores si no hay adjacentes al robot, que dos robots recogan el mismo flotador, que un robot recoja dos flotadores, flotadores fuera de rango en X e Y, sobre obstaculos, en la misma posición y que un robot pase sobre un flotador.

Por último, se define la restricción más importante, que es evitar que un robot pase sobre el agua sin un flotador, para lo que se plantea una funcion auxiliar de si es que un robot tiene flotador, que solo depende del robot R en el tiempo T, la cual se usa en la restricción que evita que las posiciones de agua y un robot sin flotador se sobrelapen.

Portales

La implementación de la mecánica de portales tiene dos partes. Primero se definen todas las combinaciones de posiciones adjacentes del portal al robot, para T=T y T=0. Luego se define la función de la acción de teletransportarse para portales adyacentes, que modifica la posición del robot a la posición de salida del portal en caso de ser adyacente a su posición de entrada y tomar la acción de teletransportarse. Al igual que antes, no se modifican lineas del código preexistente, pero si se define la acción "*teleport*" junto con las otras acciones para no generar errores.