Proiect Inteligenta Artificiala 3-Limbaje de planificare

Marcu Mihai-Alexandru

12-01-2021

1 Prezentarea temei

Tema prezentata in acest proiect este modul de functionare al unei cofetarii descris in limbajul PDDL. Aceasta vinde 6 tipuri de prajituri: fursecuri, gogosi, cheesecake, lavacake, placinta cu mere si eclere. Cofetaria se va deschide la inceputul rularii codului, dupa care clientii vor realiza cate o comanda fiecare ce va contine o lista de tipuri de prajituri dintre cele mentionate mai sus. Odata cu primirea comenzii, angajatii cofetariei vor verifica daca au ingredientele necesare pentru a pregati acele prajituri. La inceputul zilei in cofetarie nu se afla niciun ingredient. Daca ingredientele necesara prepararii unei prajituri nu se regasesc in stoc, atunci un angajat va merge la magazin si va cumpara produsul respectiv. Dupa prepararea unei prajituri din comanda, ingredientele acelui produs se vor termina din stoc. Daca ingredientele produsului cerut se regasesc in stoc, atunci se poate incepe prepararea produsului si se poate confirma comanda acestui produs. Clientul are 2 variante: poate alege ca produsele sa fie livrate la o adresa anume sau poate sa le preia chiar el. Dupa ce produsele din comanda au fost preluate de catre client, acesta va plati suma datorata cofetariei.

2 Implementare

```
(define (domain cakeshopdomain)
        (:requirements :strips :action-costs)
        (:predicates (cakeshopIsOpen)
                                (Client ?x)
                        (hasClient)
                                (cookies)
                                (donuts)
                        (cheesecake)
                        (lavacake)
                         (applepie)
                        (eclair)
                        (hasSugar)
```

```
                        (hasButter)
                        (hasFlour)
                        (hasChocolate)
                        (hasMilk)
                        (hasEggs)
                        (hasBakingPowder)
                        (hasSalt)
                         (hasCheeseCream)
                        (hasStrawberries)
                        (hasVanillaEssence)
                        (hasApples)
                        (ordersCookies ?x)
                               (ordersDonuts ?x)
                               (ordersCheesecake ?x)
                        (ordersLavacake ?x)
                        (ordersApplepie ?x)
                        (ordersEclair ?x)
                        (cookiesOrderComplete ?x)
                               (donutsOrderComplete ?x)
                              (cheesecakeOrderComplete ?x)
                        (lavacakeOrderComplete ?x)
                        (applepieOrderComplete ?x)
                        (eclairOrderComplete ?x)
                        (withDelivery ?x)
                        (productsSent ?x)
                        (clientPaid ?x))

        (:functions (total-cost))

        (:action openCakeshop
           :parameters ()
           :precondition (not (cakeshopIsOpen))
           :effect (cakeshopIsOpen))

        (:action newClient
           :parameters (?x)
           :precondition (and(not (Client ?x)) (cakeshopIsOpen) (not (hasClient)))
           :effect ( and(Client ?x) (hasClient)))

        (:action makeCookies
           :parameters ()
           :precondition (and(not(cookies)) (hasButter) (hasSugar) (hasFlour)
                   (hasBakingPowder) (hasSalt) (hasChocolate) (hasMilk))
           :effect (and(cookies) (not(hasButter)) (not(hasSugar)) (not(hasFlour))
                   (not(hasBakingPowder)) (not(hasSalt)) (not(hasChocolate)) (not(hasMilk)))
```

```
    (:action makeDonuts
       :parameters ()
       :precondition (and(not(donuts)) (hasButter) (hasSugar) (hasFlour)
              (hasBakingPowder) (hasSalt) (hasChocolate) (hasMilk) (hasEggs))
       :effect (and(donuts) (not(hasButter)) (not(hasSugar)) (not(hasFlour))
              (not(hasBakingPowder)) (not(hasSalt)) (not(hasChocolate))
              (not(hasMilk)) (not(hasEggs))))

    (:action makeCheesecake
       :parameters ()
       :precondition (and(not(cheesecake)) (hasButter) (hasSugar) (hasEggs)
              (hasBakingPowder) (hasSalt) (hasStrawberries) (hasCheeseCream))
       :effect (and(cheesecake) (not(hasButter)) (not(hasSugar)) (not(hasEggs))
              (not(hasBakingPowder)) (not(hasSalt)) (not(hasStrawberries)) (not(hasChee

    (:action makeLavacake
       :parameters ()
       :precondition (and(not(lavacake)) (hasButter) (hasSugar) (hasEggs)
              (hasFlour) (hasBakingPowder) (hasSalt) (hasChocolate) (hasMilk))
       :effect (and(lavacake) (not(hasButter)) (not(hasSugar)) (not(hasEggs))
              (not(hasFlour)) (not(hasBakingPowder)) (not(hasSalt)) (not(hasChocolate))

    (:action makeApplepie
       :parameters ()
       :precondition (and(not(applepie)) (hasButter) (hasSugar) (hasFlour)
              (hasBakingPowder) (hasSalt) (hasEggs) (hasApples))
       :effect (and(applepie) (not(hasButter)) (not(hasSugar)) (not(hasEggs))
              (not(hasFlour)) (not(hasBakingPowder)) (not(hasSalt)) (not(hasApples))))

    (:action makeEclair
       :parameters ()
       :precondition (and(not(eclair)) (hasButter) (hasSugar) (hasFlour)
              (hasBakingPowder) (hasSalt) (hasChocolate) (hasEggs) (hasMilk) (hasVanill
       :effect (and(eclair) (not(hasButter)) (not(hasSugar)) (not(hasEggs))
              (not(hasFlour)) (not(hasBakingPowder)) (not(hasSalt)) (not(hasChocolate))
              (not(hasMilk)) (not(hasVanillaEssence))))

    (:action buyButter
       :parameters ()
       :precondition (not(hasButter))
       :effect (hasButter))

    (:action buyMilk
       :parameters ()
       :precondition (not(hasMilk))
       :effect (hasMilk))
```

```
105
106         (:action buyFlour
107            :parameters ()
108            :precondition (not(hasFlour))
109            :effect (hasFlour))
110
111         (:action buySugar
112            :parameters ()
113            :precondition (not(hasSugar))
114            :effect (hasSugar))
115
116         (:action buyCheesecream
117            :parameters ()
118            :precondition (not(hasCheesecream))
119            :effect (hasCheesecream))
120
121         (:action buyChocolate
122            :parameters ()
123            :precondition (not(hasChocolate))
124            :effect (hasChocolate))
125
126         (:action buyVanillaEssence
127            :parameters ()
128            :precondition (not(hasVanillaEssence))
129            :effect (hasVanillaEssence))
130
131         (:action buyBakingPowder
132            :parameters ()
133            :precondition (not(hasBakingPowder))
134            :effect (hasBakingPowder))
135
136         (:action buySalt
137            :parameters ()
138            :precondition (not(hasSalt))
139            :effect (hasSalt))
140
141         (:action buyEggs
142            :parameters ()
143            :precondition (not(hasEggs))
144            :effect (hasEggs))
145
146         (:action buyApples
147            :parameters ()
148            :precondition (not(hasApples))
149            :effect (hasApples))
150
```

```
151         (:action buyStrawberries
152            :parameters ()
153            :precondition (not(hasStrawberries))
154            :effect (hasStrawberries))
155
156         (:action orderedCookies
157            :parameters (?x)
158            :precondition (and (Client ?x) (ordersCookies ?x) (cookies))
159            :effect (and (cookiesOrderComplete ?x) (not(cookies))
160                    (not(ordersCookies ?x))  (increase (total-cost) 1)))
161
162         (:action orderedDonuts
163            :parameters (?x)
164            :precondition (and (Client ?x) (ordersDonuts ?x) (donuts))
165            :effect ( and(donutsOrderComplete ?x) (not(donuts))
166                    (not(ordersDonuts ?x))  (increase (total-cost) 1)))
167
168         (:action orderedCheesecake
169            :parameters (?x)
170            :precondition (and (Client ?x) (ordersCheesecake ?x) (cheesecake))
171            :effect ( and(cheesecakeOrderComplete ?x) (not(cheesecake))
172                    (not(ordersCheesecake ?x))  (increase (total-cost) 2)))
173
174         (:action orderedLavacake
175            :parameters (?x)
176            :precondition (and (Client ?x) (ordersLavacake ?x) (lavacake))
177            :effect ( and(lavacakeOrderComplete ?x) (not(lavacake))
178                    (not(ordersLavacake ?x))  (increase (total-cost) 3)))
179
180         (:action orderedApplepie
181            :parameters (?x)
182            :precondition (and (Client ?x) (ordersApplepie ?x) (applepie))
183            :effect ( and(applepieOrderComplete ?x) (not(applepie))
184                    (not(ordersApplepie ?x))  (increase (total-cost) 1)))
185
186         (:action orderedEclair
187            :parameters (?x)
188            :precondition (and (Client ?x) (ordersEclair ?x) (eclair))
189            :effect ( and(eclairOrderComplete ?x) (not(eclair))
190                    (not(ordersEclair ?x))  (increase (total-cost) 2)))
191
192         (:action sendDelivery
193            :parameters (?x)
194            :precondition (and (Client ?x) (withDelivery ?x) ( or(cookiesOrderComplete ?x)
195                    (applepieOrderComplete ?x) (cheesecakeOrderComplete ?x)
196                    (donutsOrderComplete ?x) (eclairOrderComplete ?x) (lavacakeOrderComplete
```

```
            (not(or(ordersApplepie ?x) (ordersCheesecake ?x) (ordersCookies ?x)
                    (ordersDonuts ?x) (ordersEclair ?x) (ordersLavacake ?x))))
            :effect (productsSent ?x))

         (:action clientPicksProducts
            :parameters (?x)
            :precondition (and (Client ?x) ( not(withDelivery ?x)) ( or(cookiesOrderComplete
                    (applepieOrderComplete ?x) (cheesecakeOrderComplete ?x) (donutsOrderCompl
                    (eclairOrderComplete ?x) (lavacakeOrderComplete ?x))
            (not(or(ordersApplepie ?x) (ordersCheesecake ?x) (ordersCookies ?x)
                    (ordersDonuts ?x) (ordersEclair ?x) (ordersLavacake ?x))))
            :effect (productsSent ?x))

         (:action clientPays
            :parameters (?x)
            :precondition (and (Client ?x) (productsSent ?x))
            :effect ( and(clientPaid ?x) (not(hasClient)) (not (Client ?x))))
)
```

2.1 Explicatii domain:

Predicatele programului: -cakeshopIsOpen: este adevarat daca cofetaria este deschisa si fals daca aceasta este inchisa; -Client x: reprezinta clientul x; -hasClient: este adevarat daca este deja preluata o comanda sau urmatorul client poate sa plaseze comanda; -urmatoarele 6 predicate sunt adevarate daca este pregatit deja produsul cu numele respectiv sau fals in caz contrar; -urmatoarele 12 predicate reprezinta numele ingredientelor necesare pregatirii produselor si sunt adevarate daca ingredientul respectiv este in stoc; -urmatoarele 6 predicate sunt adevarate daca clientul a comandat produsul respectiv; -urmatoarele 6 predicate reprezinta confirmarea si realizarea comenzii produsului respectiv; -withDelivery x: indica daca clientul respectiv doreste ca produsele sa-i fie livrate la o adresa; -productSent x: indica daca produsele au fost primite de catre clientul x, fie ridicate de catre el, fie livrate; -clientPaid x: indica daca clientul x a platit produsele primite.

Actiunile programului: -openCakeshop: deschide cofetaria; -newClient x: se pregateste sa preia comanda unui nou client, daca acesta nu exista deja, daca cofetaria este deschisa si daca nu este preluata deja comanda altui client; -urmatoarele 6 actiuni reprezinta procesul de preparare a celor 6 prajituri, fiecare verificand daca nu sunt deja preparate produsele si daca exista in stoc ingredientele necesare pentru producerea acestora; -urmatoarele 12 actiuni reprezinta situatia in care ingredientele respective nu se afla in stoc, angajatul fiind nevoit sa le cumpere; -urmatoarele 6 actiuni indica daca comanda realizata de clientul x poate fi realizata, in caz afirmativ aceasta este confirmata, iar produsele comandate sunt scoase din lista de produse preparate deja si se taie comanda de pe lista de asteptare de comenzi; -sendDelivery x: daca comanda clientului x este realizata, indiferent de produs, si acesta nu mai are alta comanda,

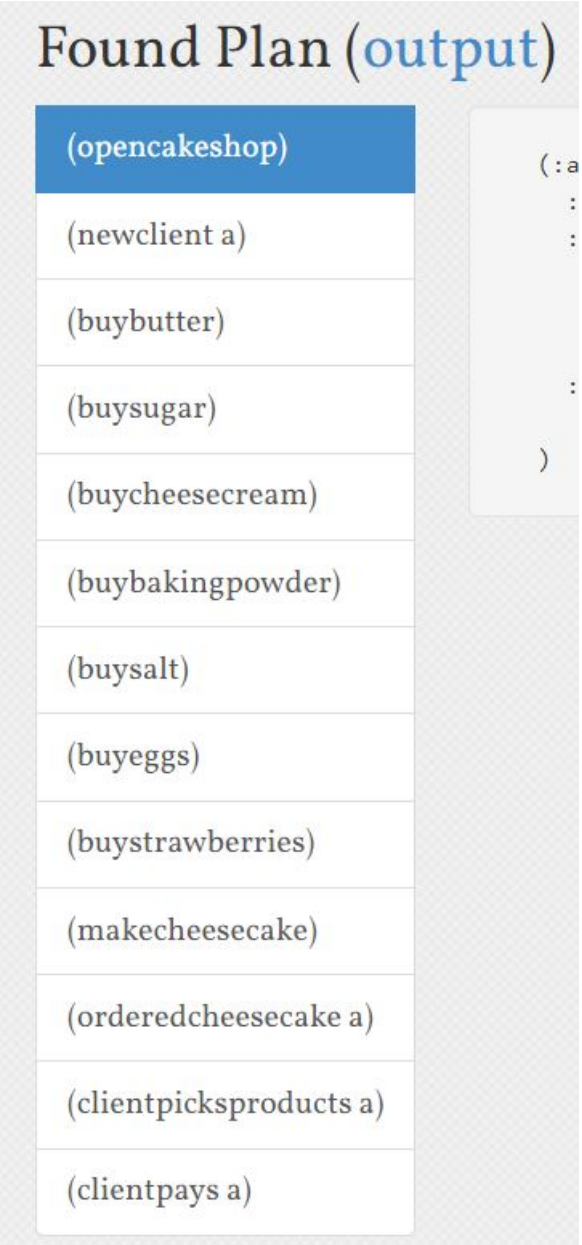
atunci produsele ii sunt livrate deoarece el a dorit aceasta optiune prin intermediul predicatului withDelivery; -clientPicksProducts x: are aceasi utilizare ca si sendDelivery, diferenta fiind faptul ca clientul x va ridica chiar el comanda; -clientulPays x: daca produsele au ajuns in posesia clientului x, atunci clientul va plati, va fi sters de pe lista de clienti si se va nega predicatul hasClient, deoarece urmatorul client isi poate plasa comanda.

3 Scenarii:

3.1 Scenariul 1:

In acest scenariu un client comanda un cheesecake fara livrare.

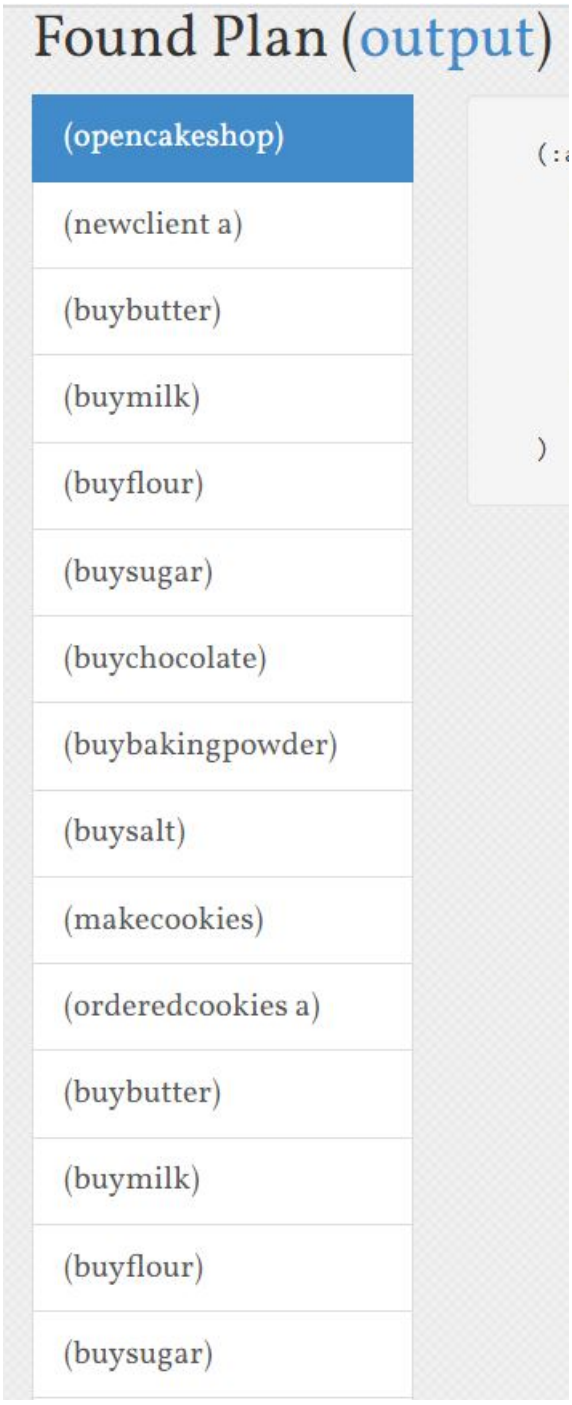
```
(define (problem cakeshopprob1)
  (:domain cakeshopdomain)
  (:objects a)
  (:init (ordersCheesecake a)
  (=(total-cost) 0))
  (:goal (clientPaid a))
  (:metric minimize(total-cost)))
```



3.2 Scenariul 2:

In acest scenariu primul client comanda un cheesecake si o portie de fursecuri impreuna cu optiunea ca acestea sa fie livrate. Un al doilea client comanda un lavacake fara livrare.

```
(define (problem cakeshopprob2)
  (:domain cakeshopdomain)
  (:objects a b)
  (:init (ordersCheesecake a) (ordersCookies a) (withDelivery a) (ordersLavacake b)
  (=(total-cost) 0))
  (:goal (and (cheesecakeOrderComplete a) (cookiesOrderComplete a) (clientPaid a)
  (lavacakeOrderComplete b) (clientPaid b)))
  (:metric minimize(total-cost)))
```


Found Plan (output)
(opencakeshop)
(newclient a)
(buybutter)
(buymilk)
(buyflour)
(buysugar)
(buychocolate)
(buybakingpowder)
(buysalt)
(makecookies)
(orderedcookies a)
(buybutter)
(buymilk)
(buyflour)
(buysugar)

(buycheesecream)

(buybakingpowder)

(buysalt)

(buyeggs)

(buystrawberries)

(makecheesecake)

(orderedcheesecake a)

(senddelivery a)

(clientpays a)

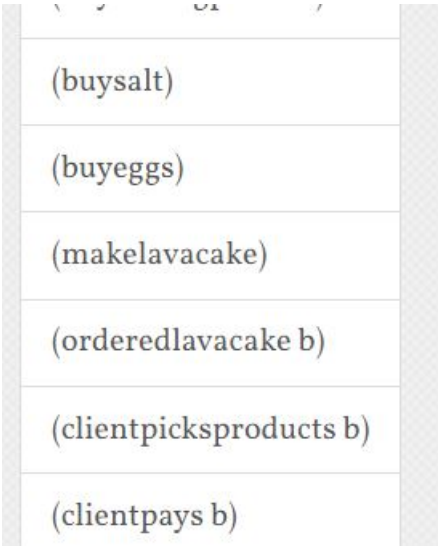
(newclient b)

(buybutter)

(buysugar)

(buychocolate)

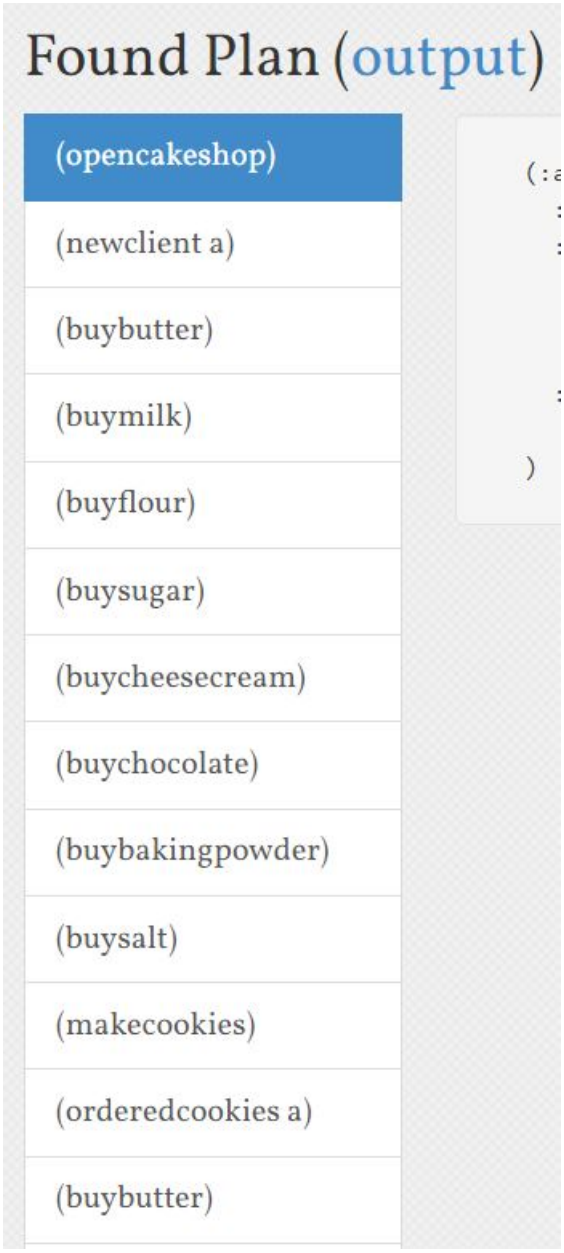
(buybakingpowder)



3.3 Scenariul 3:

In acest scenariu avem 3 clienti: primul comanda un cheesecake si o portie de fursecuri si doreste ca acestea sa fie livrate, al doilea comanda un lavacake si o portie de fursecuri cu livrare, iar cel de-al treilea client comanda o placinta cu mere si un ecler fara a dori ca acestea sa fie livrate.

```
(define (problem cakeshopprob3)
  (:domain cakeshopdomain)
  (:objects a b c)
  (:init (ordersCheesecake a) (ordersCookies a) (withDelivery a)
  (ordersLavacake b) (ordersCookies b) (withDelivery b)
  (ordersApplepie c) (ordersEclair c)
  (=(total-cost) 0))
  (:goal (and (cheesecakeOrderComplete a) (cookiesOrderComplete a) (clientPaid a)
  (lavacakeOrderComplete b) (cookiesOrderComplete b) (clientPaid b)
  (applepieOrderComplete c) (eclairOrderComplete c) (clientPaid c)))
  (:metric minimize(total-cost)))
```


Found Plan (output)
(opencakeshop)
(newclient a)
(buybutter)
(buymilk)
(buyflour)
(buysugar)
(buycheesecream)
(buychocolate)
(buybakingpowder)
(buysalt)
(makecookies)
(orderedcookies a)
(buybutter)

(buymilk)

(buyflour)

(buysugar)

(buychocolate)

(buyvanillaessence)

(buybakingpowder)

(buysalt)

(buyeggs)

(buystrawberries)

(makecheesecake)

(orderedcheesecake a)

(senddelivery a)

(clientpays a)

(newclient b)

(buybutter)

(buysugar)

(buybutter)
(buysugar)
(buybakingpowder)
(buysalt)
(makecookies)
(orderedcookies b)
(buybutter)
(buymilk)
(buyflour)
(buysugar)
(buychocolate)
(buybakingpowder)
(buysalt)
(buyeggs)
(makelavacake)
(orderedlavacake b)

(makelavacake)
(orderedlavacake b)
(senddelivery b)
(clientpays b)
(newclient c)
(buybutter)
(buymilk)
(buyflour)
(buysugar)
(buychocolate)
(buybakingpowder)
(buysalt)
(buyeggs)
(makeeclair)
(orderedeclair c)
(buybutter)

```
(buyflour)
(buysugar)
(buybakingpowder)
(buysalt)
(buyeggs)
(buyapples)
(makeapplepie)
(orderedapplepie c)
(clientpicksproducts c)
(clientpays c)
```

4 Referinte

-Indrumatorul de laborator;
-https://www.cs.toronto.edu/ sheila/2542/s14/A1/introtopddl2.pdf.