2.1 STRIPS

Imamo 3 kocke (a, b, c) in 4 možne pozicije (1,2,3,4), kot kaže slika 2.1. Robot, ki premika kocke, ima na voljo le operacijo *move*. S to akcijo lahko prime kocko na vrhu in jo premakne ter odloži na drugi kocki ali na označeni pozici na tleh. Stanja in akcijo opišemo z relacijama *clear* in *on*.

a) V STRIPS jeziku opišite začetno stanje s slike in akcijo *move*.

b) V STRIPS jeziku opišite cilj planiranja. Želimo sestaviti stolp, kjer je c spodaj, b na sredini in a na vrhu.

c) Na kratko opišite osnovne korake pri planiranju s sredstvi in cilji.

d) Simulirajte postopek planiranja. Uporabite preiskovanje v globino. Kaj je problem preiskovanja v globino?

e) Uporabite iterativno poglablanje. Pri klasičnem preiskovanju iterativno poglablanje zagotavlja optimalno rešitev. Kaj pa tu?

f) Kaj je osnovni princip regresiranja ciljev. Napišite formulo.

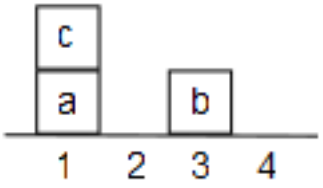
g) Simulirajte planiranje z regresijo ciljev. Uporabite iterativno poglablanje; začnite z globino $D = 3$.

Rešitev: Pri prvi nalogi bodo odgovori precej bolj podrobni, kot pri nalogah, ki sledijo. Čeprav razumevanje nekaterih rešitev zahteva malo več truda, lahko le na ta način podrobno prikažemo delovanje algoritmov planiranja.

Naloga a: opis akcije *move*

Stanje na sliki 2.1 opišemo z relacijama *clear* in *on*:

$state = [clear(2), clear(4), clear(b), clear(c), on(a, 1), on(c, a), on(b, 3)]$.



Slika 2.1: Začetno stanje v svetu kock

Za imena relacij bomo, zaradi konsistentnosti z opisi v knjigi, uporabljali angleške izraze. Z relacijo *clear* označimo prost objekt, *on* pa pomeni, da prvi objekt stoji na drugem.

Preostane nam še opis akcije *move*. Akcija ima 3 argumente:

$move(X, From, To)$,

kjer X označuje kocko, ki jo bomo premaknili, $From$ in To sta začetni in končni položaj. Akcije bomo opisali s štirimi atributi:

conditions; pogoji, ki morajo veljati v trenutnem stanju, da je akcija izvedljiva,

adds; relacije, ki jih akcija doda v stanje (temu pravimo tudi pozitivni učinki akcije),

dels; relacije, ki jih akcija izbriše iz stanja (negativni učinki),

constraints; omejitve pri določanju vrednosti spremenljivk, s katerimi lahko občutno zmanjšamo prostor preiskovanja.

Omejitve (*constraints*) in pogoji (*conditions*) imajo na prvi pogled enako vlogo. Običajno v pogoje pišemo relacije, ki jih v nadaljevanju lahko dosežemo, npr. $on(b, 2)$. V omejitvah pa so fiksni pogoji, npr. "a je kocka", in pogoji, ki preprečujejo nesmiselne akcije, npr. premik kocke na samo sebe.

Akcijo *move* lahko izvedemo pri naslednjih pogojih:

$conditions(move(X, From, To)) = [clear(X), clear(To), on(X, From)]$.

Kocka X mora biti prosta (sicer je ne moremo prijeti), To mora biti prost, da lahko kocko postavimo tja in X mora biti na $From$.

Izvedba akcije v trenutnem stanju doda relacije:

$adds(move(X, From, To)) = [clear(From), on(X, To)]$.

Objekt $From$ se po akciji sprosti in X je na mestu To. Negativni učinki akcije so: $clear(To)$ in $on(X, From)$:

$dels(move(X, From, To)) = [clear(To), on(X, From)]$.

Omejitve:

$constraints(move(X, From, To)) = [X \neq From, X \neq To, To \neq From, block(X)]$.

Omejitev $X \neq To$ preprečuje, da bi kocko X premaknili na samo sebe, $To \neq From$ pa preprečuje, da bi kocko premaknili nazaj na isto mesto. Z $block(X)$ zahtevamo, da je X kocka (sicer bi algoritem lahko poskušal premikati tudi polja 1,2,3,4).

Naloga b: opis cilja

Stolp, kjer je c spodaj in a na vrhu:

$goals = [on(a, b), on(b, c)]$.

Naloga c: princip sredstev in ciljev

Osnovni princip planiranja je sredstva-cilji (*angl.* means-ends), kjer izbiramo sredstva (akcije), ki najbolj verjetno vodijo k zadanemu cilju. Postopek ima štiri korake:

1. Izberi še nerešen cilj v $goals$. Cilje izbiramo po vrsti, kot so napisani.
2. Izberi akcijo, ki ta cilj doda v stanje. Vse akcije, ki dodajo izbran cilj v stanje, predstavljajo možnosti za nadaljevanje. Za potrebe preiskovanja akcije uredimo glede na število nerešenih predpogojev.
3. Omogoči izbrano akcijo tako, da obravnavaš njene predpogoje kot nove cilje.
4. Izvedi akcijo, ki doda izbran cilj v trenutno stanje.
5. Če obstajajo nerešeni cilji, se vrni na korak 1.

Za preiskovanje lahko uporabimo poljuben algoritem; v globino, iterativno poglabljanje, A*, itd.

Naloga d: preiskovanje v globino

V naši domeni sveta kock bi postopek potekal takole:

1. Izberemo cilj $on(a, b)$.

2. Za cilj poiščemo akcijo, ki ga vzpostavi. To informacijo vsebuje množica $adds$. Akcija, ki doda relacijo $on(a, b)$ je: $move(a, From, b)$. Predpogoji zanjo so:

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Zaradi nedoločene vrednosti $From$ imamo 5 možnih akcij: $move(a, 1, b)$, $move(a, 2, b)$, $move(a, 3, b)$, $move(a, 4, b)$ in $move(a, c, b)$. Zaradi omejitev a in b nista dovoljeni vrednosti spremenljivke $From$. Najprej poskusimo s $From = 1$, ker sta potem dva predpogoja že izpolnjena v začetnem stanju.

3. Predpogoji $[clear(a), clear(b), on(a, 1)]$ postanejo trenutni cilji in rekurzivno kličemo program za planiranje (zato znak ' pri oznaki koraka).

1'. Izberemo cilj $clear(a)$.

2'. Akcija: $move(X, a, To)$. Torej, nekaj moramo premakniti iz a in postaviti drugam, da bo a prost. Predpogoji za to akcijo so:

$conditions(move(X, a, To)) = \{clear(X), clear(To), on(X, a)\}$.

Vrednosti za X in To, izpolnjene že v začetnem stanju, so $X = c$ in $To = 2$.

3'. Korak ni potreben, vsi pogoji so izpolnjeni v trenutnem stanju.

4'. Izvedemo akcijo

$$1.move(c, a, 2)$$

in dobimo novo stanje tako, da iz stanja *initial_state* izbrišemo vse iz *dels*: $clear(2), on(c, a)$ in dodamo vse iz *adds*: $clear(a), on(c, 2)$. Novo stanje je

$$state = [clear(4), clear(a), clear(b), clear(c), on(a, 1), on(c, 2), on(b, 3)].$$

5'. Vsi cilji (predpogoji) so izpolnjeni, zaključimo.

3. Zdaj so izpolnjeni vsi predpogoji za akcijo $move(a, 1, b)$.

4. Izvedemo akcijo:

$$2.move(a, 1, b)$$

in dobimo stanje:

$$state = [clear(1), clear(4), clear(a), clear(c), on(a, b), on(c, 2), on(b, 3)].$$

5. Cilj $on(b, c)$ še ni izpolnjen. Vrnemo se na korak 1.

1. Izpolnili smo prvi cilj $on(a, b)$ in se lotimo drugega, $on(b, c)$.

2. Ta cilj dosežemo z akcijo $move(b, From, c)$, ki ima predpogoje:

$$conditions(move(b, From, c)) = [clear(b), clear(c), on(b, From)].$$

Izberemo $From = 3$.

3. Rešujemo predpogoje $[clear(b), clear(c), on(b, 3)]$.

1'. Izberemo cilj $clear(b)$.

2'. Akcija: $move(X, b, To)$ in njeni predpogoji so

$$conditions(move(X, b, To)) = [clear(X), clear(To), on(X, b)].$$

Že izpolnjene vrednosti za X in To so $X = a$ in $To = 1$.

3'. Vsi pogoji so izpolnjeni.

4'. Izvedemo akcijo:

$$3.move(a, b, 1)$$

in dobimo stanje

$$state = [clear(4), clear(a), clear(b), clear(c), on(a, 1), on(c, 2), on(b, 3)].$$

5'. Vsi cilji so izpolnjeni, zaključimo.

3 in 4. Predpogoji so izpolnjeni in izvedemo četrto akcijo, s katero dosežemo $on(b, c)$:

$$4.move(b, 3, c)$$

in dobimo stanje

$$state = \{clear(3), clear(4), clear(a), clear(b), on(a, 1), on(c, 2), on(b, c)\}.$$

5. Med reševanjem $on(b, c)$ smo podrli cilj $on(a, b)$ in ga moramo ponovno reševati. Vrnemo se na korak 1.

1. Cilj: $on(a, b)$

2. Akcija: $move(a, From, b)$ izpolni ta cilj in, ker predpogoji zanjo pri $From = 1$ v danem stanju veljajo, jo lahko (4. korak) izvedemo:

$$5.move(a, 1, b)$$

Končno stanje je:

$state = [clear(1), clear(3), clear(4), clear(a), on(a, b), on(c, 2), on(b, c)]$.

5. Vsi cilji so doseženi, končamo postopek.

S planiranjem smo našli rešitev:

1. $move(c, a, 2)$,
2. $move(a, 1, b)$,
3. $move(a, b, 1)$,
4. $move(b, 3, c)$,
5. $move(a, 1, b)$.

Rešitev ni optimalna, saj je problem rešljiv v le treh potezah. V nadaljevanju si bomo pogledali, kako lahko najdemo krajše rešitve z a) uporabo interativnega poglabljanja in b) regresijo ciljev.

Naloga e: preiskovanje z iterativnim poglabljanjem

Pri iterativnem poglabljanjem bomo postopoma povečevali dolžino D dovoljenega plana. Zaradi preglednosti bomo v simulaciji nekaj korakov izpustili.

Imamo isti cilj:

$goals = [on(a, b), on(b, c)]$

v naši začetni poziciji:

$initial\_state =$
$[clear(2), clear(4), clear(b), clear(c), on(a, 1), on(c, a), on(b, 3)]$.

$$D = 1$$
(največja dolžina plana je 1)

(d=0) Rešujemo cilje $on(a,b), on(b,c)$**.**

Izberemo cilj $on(a,b)$. Akcija, ki doda $on(a,b)$ je $move(a, From, b)$. Predpogoji so:

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Ker $clear(a)$ v začetnem stanju ni resničen, potrebujemo še eno akcijo. Cilja torej ne moremo rešiti samo z eno akcijo.

Izberemo cilj $on(b,c)$. Akcija: $move(b, From, c)$, predpogoji so:

$conditions(move(b, From, c)) = [clear(b), clear(c), on(b, From)]$.

Možne vrednosti za $From$ so: $3, 1, 2, 4, a$. Pri vrednosti $From = 3$ lahko izvedemo akcijo $move(b, 3, c)$ in dosežemo $on(b,c)$. V tem stanju še vedno ni dosežen $on(a,b)$, torej to ni rešitev. Vse druge vrednosti za $From$ prav tako zahtevajo dodatne akcije za uresničitev predpogojev in jih zaradi omejitve $D = 1$ ne moremo izpolniti. Pri $D = 1$ torej ne moremo najti rešitve.

$$D = 2$$

(d=0) Rešujemo cilje $on(a,b), on(b,c)$**.**

Akcija: $move(a, From, b)$, predpogoji so

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Možne vrednosti za $From$ so: $1, 2, 3, 4, c$. Nastavimo $From = 1$, rešiti moramo:

$[clear(a), clear(b), on(a, 1)]$.

(d=1) Rešujemo cilje $[clear(a), clear(b), on(a, 1)]$

Zdaj smo na globini 1 in lahko uporabimo le še eno akcijo, saj bomo v vsakem primeru izvedli $move(a, 1, b)$. Izberemo cilj $clear(a)$, akcija: $move(X, a, To)$. Predpogoji so:

$conditions(move(X, a, To)) = [clear(X), clear(To), on(X, a)]$,

kjer so možne vrednosti za X in To:

$$(c,2), (c,4), (c,b), (c,1), (c,3), (b,2), (b,4), (b,c), (b,1), (b,3).$$

Prvi trije nabori $(c,2), (c,4), (c,b)$ izpolnjujejo vse pogoje in so zato na začetku, $(c,1), (c,3)$ izpolnjujeta le dva pogoja, itd.

Izberimo $X = c$ in $To = 2$. Zdaj lahko izvedemo prvo akcijo $1.move(c, a, 2)$ in takoj nato še akcijo $2.move(a, 1, b)$ iz prejšnjega koraka. Dobimo stanje:

$state = [clear(1), clear(4), clear(a), clear(c), on(a, b), on(c, 2), on(b, 3)]$.

V tem stanju ni izpolnjen cilj $on(b, c)$ in ker smo naredili že dva koraka, se vrnemo in poskusimo drugo pot.

Postopek bi nadaljevali z naslednjim naborom vrednosti za X in To. Hitro opazimo, da z uporabo zgoraj naštetih vrednosti ne bi našli rešitve in se moramo vrniti na globino d=0.

(d=0) Rešujemo cilje $on(a, b), on(b, c)$, nadaljevanje.

V množici ciljev $[clear(a), clear(b), on(a, 1)]$ smo poskušali rešiti $clear(a)$, a nam ni uspelo priti do rešitve. Preostala dva cilja sta že izpolnjena, zato jih nima smisla reševati. Poskusimo naslednjo nastavitev za $From$; $From = 2$. Nov nabor ciljev je:

$$[clear(a), clear(b), on(a, 2)].$$

Najprej bi poskusili še enkrat rešiti $clear(a)$ z istim rezultatom kot zgoraj. Potem bi poskušali reševati $on(a, 2)$, a brez uspeha, saj za premik kocke a na drugo mesto potrebujemo $clear(a)$. Enako se zgodi pri ostalih vrednostih $From$.

Ne preostane nam drugega, kot da izberemo cilj: $on(b, c)$. Akcija je: $move(b, From, c)$, predpogoji pa:

$conditions(move(b, From, c)) = [clear(b), clear(c), on(b, From)]$.

Možne vrednosti za $From$ so: $3, 1, 2, 4, a$. Nastavimo $From = 3$ in dobimo množico ciljev:

$[clear(b), clear(c), on(b, 3)]$.

Vsi cilji so izpolnjeni, torej so izpolnjeni predpogoji za $1.move(b, 3, c)$.

Algoritem bi zdaj izračunal novo stanje in potem nadaljeval z reševanjem $on(a, b)$. Ugotovil bi, da cilj ni rešljiv z eno akcijo. Nato bi se vrnil in poskušal s $From = 1$, kar pomeni, da bi najprej premaknil kocko b na polje 1 in jo potem dal na c. Spet se ne bi izšlo. To bi ponavljal še s preostalimi vrednostmi za $From$ in vendar neuspel. S tem bi se zaključilo preiskovanje pri $D = 2$.

$$D = 3$$

Vemo, da je problem rešljiv s tremi premiki. Torej bi rešitev morali najti pri $D = 3$. Pa jo res?

(d=0) Rešujemo cilje $on(a,b), on(b,c)$.

Izberemo cilj $on(a,b)$, akcija: $move(a, From, b)$, predpogoji so

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Možne vrednosti za $From$ so: $1, 2, 3, 4, c$. Nastavimo $From = 1$ in dobimo novo množico ciljev:

$[clear(a), clear(b), on(a, 1)]$.

(d=1) Rešujemo cilje $[clear(a), clear(b), on(a, 1)]$.

Izberemo cilj $clear(a)$, akcija: $move(X, a, To)$. Predpogoji so:

$conditions(move(X, a, To)) = [clear(X), clear(To), on(X, a)]$,

kjer so možne vrednosti za X in To: $(c, 2), (c, 4), (c, b), ...$ Izvedemo akciji $1.move(c, a, 2)$ in $2.move(a, 1, b)$, kot pri $D = 2$. Zdaj ostane še reševanje $on(b, c)$.

Očitno $on(b, c)$ ne dosežemo z eno akcijo, zato postopka ne bomo nadaljevali. Vprašanje je, ali je možno doseči rešitev po drugi poti; npr. če bi v koraku A spremenili vrednost $From$? Izkaže se da ne, saj algoritem poskuša najprej doseči $on(a, b)$, a pri tem ne uspe, saj bi moral najprej poskrbeti za $on(b, c)$.

(d=0) Rešujemo cilje $on(a,b), on(b,c)$, **nadaljevanje.**

Izberemo $on(b, c)$, akcija: $move(b, From, c)$ in nastavimo $From = 3$, kot pri $D = 2$. Dobimo cilje:

$[clear(b), clear(c), on(b, 3)]$.

Vsi cilji so izpolnjeni, izvedemo akcijo $1.move(b, 3, c)$. Spet hitro vidimo, da ne bomo uspeli z dvema akcijama, saj potrebujemo dva koraka za dosego $clear(a)$, ki je predpogoj za akcijo, ki bi dosegla $on(a, b)$.

$$D = 4$$

(d=0) Rešujemo cilje $on(a,b), on(b,c)$.

Izberemo cilj $on(a, b)$. Akcija: $move(a, From, b)$, predpogoji so

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Možne vrednosti za $From$ so: $1, 2, 3, 4, c$. Nastavimo $From = 1$ in dobimo cilje:

$[clear(a), clear(b), on(a, 1)]$.

(d=1) Rešujemo cilje $[clear(a), clear(b), on(a, 1)]$.

Izberemo cilj $clear(a)$. Akcija: $move(X, a, To)$. Predpogoji so:

$conditions(move(X, a, To)) = [clear(X), clear(To), on(X, a)]$,

vrednosti za X in To so:

$$(c, 2), (c, 4), (c, b), (c, 1), (c, 3), (b, 2), (b, 4), (b, c), (b, 1), (b, 3).$$

Pri izbranih vrednostih $(c, 2)$ lahko izvedemo akciji $move(c, a, 2)$ in $move(a, 1, b)$ in dosežemo stanje:

$state = [clear(1), clear(4), clear(a), clear(c), on(a, b), on(c, 2), on(b, 3)]$.

Ostane še reševanje $on(b, c)$. S preostalima dvema akcijama nam to ne uspe, saj moramo najprej umakniti a iz b, potem premakniti b na c in še enkrat a na b. Vrnimo se in poskusimo z drugimi vrednostmi za X in To.

Pri $(c, 4)$,(c, b),$(c, 1)$,$(c, 3)$,$(b, 2)$ in $(b, 4)$ naletimo na podoben problem. Šele pri $X = b$ in $To = c$ najdemo rešitev. Naša akcija je torej $move(b, a, c)$, predpogoji so:

$conditions(move(b, a, c)) = [clear(b), clear(c), on(b, a)]$,

(d=2) Rešujemo cilje $[clear(b), clear(c), on(b, a)]$.

Izberemo $on(b, a)$, akcija: $move(b, From, a)$, predpogoji:

$conditions(move(b, From, a)) = [clear(b), clear(a), on(b, From)]$,

Poskusimo s $From = 3$, predpogoji za akcijo $move(b, 3, a)$ so:

$[clear(b), clear(a), on(b, 3)]$.

(d=3) Rešujemo cilje $[clear(b), clear(a), on(b, 3)]$.

Izberemo $clear(a)$. Akcija: $move(X, a, To)$. Predpogoji so:

$conditions(move(X, a, To)) = [clear(X), clear(To), on(X, a)]$,

vrednosti za X in To so:

$$(c, 2), (c, 4), (c, b), (c, 1), (c, 3), (b, 2), (b, 4), (b, c), (b, 1), (b, 3).$$

Tokrat rešitev najdemo že kar pri prvem naboru vrednosti $(c, 2)$. Zdaj lahko izvedemo

$$1.move(c, a, 2)$$

in dobimo novo stanje:

$state = [clear(4), clear(a), clear(b), clear(c), on(a, 1), on(b, 3), on(c, 2)]$.

(d=2) Rešujemo cilje $[clear(b), clear(c), on(b, a)]$, **nadaljevanje.**

Predpogoji $[clear(b), clear(a), on(b, 3)]$ so izpolnjeni, lahko izvedemo akcijo

$$2.move(b, 3, a).$$

Novo stanje:

$state = [clear(3), clear(4), clear(b), clear(c), on(a, 1), on(b, a), on(c, 2)].$

(d=1) Rešujemo cilje $[clear(a), clear(b), on(a, 1)]$, **nadaljevanje.**

Izpolnjeni predpogoji $[clear(b), clear(c), on(b, a)]$, izvedemo akcijo:

$$3.move(b, a, c)$$

Novo stanje:

$state = [clear(3), clear(4), clear(a), clear(b), on(a, 1), on(b, c), on(c, 2)].$

(d=0) Rešujemo cilje $on(a, b), on(b, c)$.

Izpolnjeni predpogoji $[clear(a), clear(b), on(a, 1)]$, izvedemo akcijo:

$$3.move(a, 1, b)$$

Novo stanje:

$state = [clear(1), clear(3), clear(4), clear(a), on(a, b), on(b, c), on(c, 2)].$

Oba cilja sta dosežena, zaključimo postopek!

Našli smo rešitev:

1. $move(c, a, 2)$,
2. $move(b, 3, a)$,
3. $move(b, a, c)$,
4. $move(a, 1, b)$.

Ta naloga kaže, da je planiranje zelo povezano s preiskovanjem. Mnogokrat se namreč zgodi, da prva izbrana akcija, ki cilj uresniči, ni primerna. V koraku z $d = 1$ smo šele v sedmem poskusu z akcijo $move(b, a, c)$ prišli do cilja.

Naloga f: planiranje z regresijo ciljev

Pri metodi sredstva-cilji se posamezni cilji rešujejo lokalno, kar mnogokrat onemogoča poiskati najkrajšo rešitev. To imenujemo Sussmanova anomalija. Regresiranje ciljev se problema loti globalno in poskuša doseči vse cilje naenkrat. S tem omogoča iskanje optimalnih planov.

Postopek začnemo tako, da iz množice ciljev izberemo cilj in ustrezno akcijo, s katero bi ta cilj dosegli. V naslednjem koraku se vprašamo, kaj vse bi moralo veljati, da bi bili po izvedeni izbrani akciji doseženi vsi cilji (ne samo izbrani). Odgovor na to vprašanje je v regresiji ciljev, ki iz prejšnjih ciljev, pogojev za akcijo in učinkov akcije izračuna nove cilje. Od tu naprej nas zanimajo le ti novi cilji, saj če jih uresničimo, bomo na koncu z izbrano akcijo rešili vse začetne cilje. Postopek se rekurzivno ponavlja dokler ne dobimo množice ciljev, ki so izpolnjeni že v začetni poziciji.

Algoritem (na nivoju i; na začetku so cilji $goals(0)$ enaki končnim ciljem):

1. Če so vsi cilji v $goals(i)$ resnični v začetnem stanju, končamo in vse akcije izvedemo v obratnem vrstnem redu. Če $goals(i)$ ni možno doseči (nemogoči cilji ali ni primerne akcije), se vrnemo v prostoru stanj in poskušamo najti rešitev po drugi poti.

2. Če cilji $goals(i)$ še niso uresničeni in so izvedljivi, izberemo cilj iz $goals(i)$ in akcijo A, ki doda ta cilj in regresiramo cilje po naslednji formuli:

$$goals(i+1) = goals(i) \cup conditions(A) \setminus adds(A)$$

Pri tem moramo paziti, da A ne izbriše trenutnega cilja (v $del(A)$ ni cilja iz $goals(i)$, oz. presek $del(A)$ in $goals(i)$ je prazna množica).

Naloga g: simuliranje algoritma z regresiranjem ciljev

Za zagotavljanje najkrajše rešitve uporabimo iterativno poglabljanje. Začnemo pri $D = 3$, globini $D = 1$ in $D = 2$ zaradi preglednosti izpustimo.

$goals(0) = [on(a, b), on(b, c)]$

Ali so cilji $on(a, b), on(b, c)$ izpolnjeni v začetni poziciji? Ne. Izberemo cilj: $on(a, b)$. Akcija: $move(a, From, b)$, predpogoji so:

$conditions(move(a, From, b)) = [clear(a), clear(b), on(a, From)]$.

Možne vrednosti za $From$ so: $1, 2, 3, 4, c$. Nastavimo $From = 1$.

$adds(move(a, 1, b)) = [clear(1), on(a, b)]$
$dels(move(a, 1, b)) = [clear(b), on(a, 1)]$

Akcijo lahko izvedemo, ker $dels$ ne vsebuje cilja iz $goals(0)$.

Regresija ciljev:

$goals(1) = goals(0) \cup conditions(move(a,1,b)) \setminus adds(move(a,1,b)) =$
$= [on(a,b), on(b,c)] \cup [clear(a), clear(b), on(a,1)] \setminus [clear(1), on(a,b)] =$
$= [clear(a), clear(b), on(a,1), on(b,c)].$

Regresirane cilje interpretiramo takole: če lahko pridemo v stanje, kjer velja $goals(1)$, bomo z akcijo $move(a,1,b)$ prišli v stanje, kjer velja $goals(0)$. Postopek nadaljujemo, dokler $goals(i)$ niso izpolnjeni v začetnem stanju.

$$goals(1) = [clear(a), clear(b), on(a,1), on(b,c)]$$

Ali so novi cilji $goals(1)$ izpolnjeni v začetni poziciji? Ne. Izberemo cilj: $clear(a)$ iz $goals(1)$ (po vrsti) in izberemo akcijo $move(X,a,To)$. Predpogoji so:

$$conditions(move(X,a,To)) = [clear(X), clear(To), on(X,a)],$$

možne vrednosti za X in To so: $(c,2)$,$(c,4)$, (c,b), $(c,1)$, itd. Izberemo $X = c$, $To = 2$.

$adds(move(c,a,2)) = [clear(a), on(c,2)]$
$dels(move(c,a,2)) = [clear(2), on(c,a)]$

Akcijo lahko izvedemo, ker $clear(2)$ in $on(c,a)$ nista v trenutni množici ciljev $goals(1)$.

Regresija ciljev:

$goals(2) = goals(1) \cup conditions(move(c,a,2)) \setminus adds(move(c,a,2)) =$
$= [clear(a), clear(b), on(a,1), on(b,c)] \cup [clear(c), clear(2), on(c,a)] \setminus$
$[clear(a), on(c,2)] =$
$= [clear(c), clear(2), on(c,a), clear(b), on(a,1), on(b,c)].$

Tu ne moremo nadaljevati, saj cilj ni izvedljiv! Hkrati ni možno doseči ciljev $on(b,c)$ in $clear(c)$.

Zdaj lahko poskusimo z drugimi vrednostmi spremenljivk X in To, vendar ne bi našli rešitve v treh ali manj korakih. Vrnemo se korak nazaj; izbrati moramo nov cilj.

Cilja $clear(b)$ in $on(a,1)$ v $goals(1)$ sta v začetnem stanju že resnična, izberemo cilj: $on(b,c)$. Akcija $move(b,From,c)$, predpogoji:

$$conditions(move(b,From,c)) = [clear(b), clear(c), on(b,From)].$$

Možne vrednosti za $From$ so: $3, 1, 2, 4, a$. Nastavimo $From = 3$.

$adds(move(b,3,c)) = [clear(3), on(b,c)]$
$dels(move(b,3,c)) = [clear(c), on(b,3)]$

V *dels* ni relacije, ki bi bila tudi v trenutnih ciljih, torej lahko izvedemo akcijo.

Regresija ciljev:

$goals(2) = goals(1) \cup conditions(move(b,3,c)) \setminus adds(move(b,3,c)) =$
$= [clear(a), clear(b), on(a,1), on(b,c)] \cup [clear(b), clear(c), on(b,3)] \setminus$
$[clear(3), on(b,c)] = = [clear(a), clear(b), clear(c), on(a,1), on(b,3)].$

$$goals(2) = [clear(a), clear(b), clear(c), on(a,1), on(b,3)]$$

Ali so cilji $goals(2)$ resnični v začetnem stanju? Ne. Izberemo cilj: $clear(a)$, izberemo akcijo $move(X,a,To)$, njeni predpogoji so:

$$conditions(move(X,a,To)) = [clear(X), clear(To), on(X,a)],$$

vrednosti za X in To so: $(c,2), (c,4), (c,b), ...$ Nastavimo vrednosti $X = c$, $To = 2$.

$adds(move(c,a,2)) = [clear(a), on(c,2)]$
$dels(move(c,a,2)) = [clear(2), on(c,a)]$

Med cilji v $goals(2)$ in $dels(move(c,a,2))$ ni konflikta .

Regresija ciljev:

$goals(3) = goals(2) \cup conditions(move(c,a,2)) \setminus adds(move(c,a,2)) =$
$= [clear(a), clear(b), clear(c), on(a,1), on(b,3)] \cup$
$[clear(c), clear(2), on(c,a)] \setminus [clear(a), on(c,2)] =$
$= [clear(b), clear(c), on(a,1), on(b,3), clear(2), on(c,a)].$

$$goals(3) = [clear(b), clear(c), on(a,1), on(b,3), clear(2), on(c,a)]$$

Opazimo, da so ti cilji resnični v začetnem stanju. Zdaj le še izvedemo akcije v obratnem vrstnem redu. Rešitev je:

1. $move(c,a,2)$
2. $move(b,3,c)$
3. $move(a,1,b)$

2.2 Razširjen svet kock

V tej nalogi bomo svet kock iz prejšnje naloge razširili z dodatnimi akcijami.

a) Predpostavimo, da lahko robot porine celoten stolp levo ali desno. Pri tem mora imeti prosto ustrezno polje (levo oz. desno), kamor premika stolp. Opišite akciji $pushLeft$ in $pushRight$, ki predstavljata levi in desni premik.

b) Imejmo spretnega robota, ki zna zagrabiti dve zgornji kocki na istem stolpcu in ju zamenjati. Definirajte akcijo $swap(X, Y, From)$, ki kocki zamenja.

c) Robot lahko prime dve kocki in ju odnese na drugo polje. Definirajte akcijo $move2(X, Y, From, To)$.

d) Kako bi posodobljen robot z novimi akcijami reševal problem opisan na začetku tega poglavja? Napišite vse možne akcije, s katerimi bi poskušal pri prvem cilju $on(a, b)$? Kakšni so pogoji za izbrane akcije? Določite smiselne vrednosti spremenljik, ki naj jih algoritem najprej poskusi. Uporabljajte planiranje brez regresije ciljev.

e) Poišči množico regresiranih ciljev $rgoals$ skozi akcije $move(a, 1, b)$, $swap(b, a, 1)$ in $move2(c, a, 3, b)$, začetni cilji so $goals = [on(a, b), on(b, c)]$. Rešujemo cilj $on(a, b)$. Ali lahko akcije izvedemo?

Rešitev:

Naloga a: premikanje levo in desno

Akcija: $pushLeft(X, From, To)$
$conditions = [on(X, From), clear(To)]$
$adds = [on(X, To), clear(From)]$
$dels = [on(X, From), clear(To)]$
$constraints = [not\ block(To), not\ block(From), From = To + 1, From > 1]$

Akcija: $pushRight(X, From, To)$
$conditions = [on(X, From), clear(To)]$
$adds = [on(X, To), clear(From)]$
$dels = [on(X, From), clear(To)]$
$constraints = [not\ block(To), not\ block(From), From = To - 1, From < 4]$

Naloga b: zamenjava dveh kock

Akcija: $swap(X, Y, From)$
$conditions = [on(X, Y), clear(X), on(Y, From)]$

$adds = [on(Y, X), clear(Y), on(X, From)]$
$dels = [on(X, Y), clear(X), on(Y, From)]$
$constraints = [block(X), block(Y)]$

Naloga c: premikanje dveh kock

Akcija: $move2(X, Y, From, To)$
$conditions = [on(X, Y), clear(X), clear(To), on(Y, From)]$
$adds = [clear(From), on(Y, To)]$
$dels = [clear(To), on(Y, From)]$
$constraints = [X \neq Y, X \neq From, Y \neq From, X \neq To, Y \neq To, To \neq From, block(X), block(Y)]$

Naloga d: planiranje

Poskušal bi z vsemi akcijami, ki dodajo relacijo $on(a, b)$. To so akcije $move$, $swap$ in $move2$. Akciji $pushLeft$ in $pushRight$ nista primerni, čeprav imata med svojimi pozitivnimi učinki $on(X, To)$, saj drugi argument ne sme biti kocka.

Akcija $move(a, From, b)$, pogoji za akcijo so:

$[clear(a), clear(b), on(a, From)]$

Algoritem bi najprej poskusil s $From = 1$, saj je a v začetni poziciji na mestu 1.

Akcija $swap(b, a)$, pogoji so: $[on(b, a), clear(b)]$.

Akcija $move2(X, b, From, a)$, pogoji: $[on(X, b), clear(X), clear(a), on(b, From)]$. Najprej bi izbral vrednosti ($X = c, From = 3$). $From = 3$ bi izbral, ker je b na začetku na 3, c, ker je edina preostala kocka (X ne more biti ne a in ne b).

Naloga e: planiranje z regresijo ciljev

1. $move(a, 1, b)$
$rgoals = [clear(a), clear(b), on(a, 1), on(b, c)]$

Če dosežemo $rgoals$, bomo z akcijo $move(a, 1, b)$ dosegli $goals$.

2. $swap(b, a, 1)$
$rgoals = [on(b, a), on(a, 1), clear(b), on(b, c)]$

Akcije ne bomo mogli izvesti, saj so cilji neizvedljivi! Kocka b ne more biti hkrati na a in c.

3. $move2(c, a, 3, b)$
$rgoals = [on(c, a), clear(c), clear(b), on(a, 3), on(b, c)]$

2.3 Falcon

Falcon je vladno letalo[1], ki se ne uporablja prav dosti. Naš cilj bo narediti načrt leta, da bi s čim manj leti prepeljali slovenske ministre po evropskih državah. Na letalu je hkrati lahko poljubno število ministrov, a so ministri včasih skregani in takrat nočejo leteti skupaj. Skregana ministra ne smeta biti hkrati na letalu.

a) Opis akcij: V domeni imamo tri možne akcije: fly (premakne Falcona iz ene države v drugo), *embark* (vkrcanje ministra) in *disembark* (izkrcanje ministra). Opišite vse akcije z jezikom STRIPS. Na voljo imate relacije $incountry(Where, X)$, ki označuje, da je X v državi $Where$, $country(Where)$ vrne True, če je X država, $minister(X)$, če je X minister, $dislikes(X, Y)$ je True, če se ministra X in Y ne marata, in $onplane(X)$ označuje, da je minister X na letalu.

b) Planiranje Falcona: Imejmo začetno stanje: $\{incountry(france, m1), incountry(slovenia, falcon)\}$ in cilje $goals = [incountry(slovenia, m1), incountry(slovenia, falcon)]$. Rešite problem s planiranjem po principu sredstva-cilji in pri tem uporabite regresiranje ciljev. Poskusite nalogo rešiti s ščitenjem ciljev in brez. Ščitenje ciljev (*angl.* protecting goals) je preprečevanje algoritmu planiranja, da bi "porušil" že dosežene cilje.

c) Naj bo začetno stanje $[incountry(slovenia, m1), incountry(slovenia, falcon), onplane(m2), onplane(m3), dislikes(m2, m1)]$ in cilj je $onplane(m1)$. Zakaj s planiranjem ne moremo doseči cilja? Kako bi problem odpravili?

Rešitev:

Naloga a: opis akcij

Akcija: $fly(From, To)$
$conditions = [incountry(From, falcon)]$
$adds = [incountry(To, falcon)]$
$dels = [incountry(From, falcon)]$
$constraints = [country(From), country(To)]$

Akcija: $embark(Where, Who)$
$conditions = [incountry(Where, falcon), incountry(Where, Who)]$
$adds = [onplane(Who)]$
$dels = [incountry(Where, Who)]$

[1] V času pisanja te skripte je slovenska vlada vsekakor še imela to letalo.

$constraints =$
$[land(Where), minister(Who), not\ dislikes(Who, X), onplane(X)]$

Akcija: $disembark(Where, Who)$
$conditions = [incountry(Where, falcon), onplane(Who)]$
$adds = [incountry(Where, Who)]$
$dels = [onplane(Who)]$
$constraints = [land(Where), minister(Who)]$

Dodatno vprašanje: ali bi znali to predstavitev posplošiti za več vladnih letal?

Naloga b: planiranje poletov s Falconom

S ščitenjem ciljev. Izberimo cilj: $incountry(slovenia, m1)$. To relacijo doda le akcija $disembark(slovenia, m1)$, ki ima predpogoje:

$conditions(disembark(slovenia, m1)) =$
$[incountry(slovenia, falcon), onplane(m1)]$

Regresirani cilji so $[incountry(slovenia, falcon), onplane(m1)]$.

Izberemo cilj $onplane(m1)$, ki še ni izpolnjen. Akcija $embark(Where, m1)$ s predpogoji

$conditions(embark(Where, m1)) =$
$[incountry(Where, falcon), incountry(Where, m1)]$

Za $Where$ bi bilo tu najbolje vzeti $france$, saj se minister $m1$ tam nahaja. Problem je, da potem dobimo neizvedljive regresirane cilje

$[incountry(france, falcon), incountry(france, m1), incountry(france, falcon)$.

Letalo ne more biti hkrati na dveh mestih.

Alternativna možnost je, da izberemo cilj $incountry(slovenia, falcon)$, vendar to s ščitenjem ciljev ni možno. S ščitenjem ciljev ne najdemo rešitve.

Brez ščitenja ciljev. Izberemo cilj $incountry(slovenia, falcon)$ in akcijo: $fly(From, slovenia)$ s predpogoji:

$conditions(fly(From, slovenia)) = [incountry(From, falcon)]$

$From$ je lahko katerakoli država, rešitev bomo našli pri $From = france$.

Regresirani cilji so $[incountry(france, falcon), onplane(m1)]$

Izberemo cilj $onplane(m1)$. Akcija $embark(Where, m1)$, kjer je $Where = france$. Novi regresirani cilji so $[incountry(france, falcon), incountry(france, m1)]$.

Preostane nam le še cilj $incountry(france, falcon)$. Akcija: $fly(From, france)$. Pri vrednosti $From = slovenia$ lahko akcijo izvedemo v začetnem stanju. Rezultat planiranja je:
$fly(slovenia, france)$,
$embark(france, m1)$,
$fly(france, slovenia)$,
$disembark(slovenia, m1)$.

Naloga c

Cilj $onplane(m1)$. Akcija: $embark(slovenia, m1)$

$conditions = [incountry(slovenia, falcon), incountry(slovenia, m1)]$

Pogoji so izpolnjeni, a akcije ne moremo izvesti zaradi omejitev
$not\ dislikes(Who, X), onplane(X)$.

Omejitev bi morali zapisati kot pogoj. Potem bi pogoji ne bili izpolnjeni in algoritem planiranja bi jih poskusil izpolniti. Opis akcije *embark* spremenimo v:

Akcija: $embark(Where, Who)$
$conditions = [incountry(Where, falcon), incountry(Where, Who),$
$dislikes(Who, X), not\ onplane(X)]$
$adds = [onplane(Who)]$
$dels = [incountry(Where, Who)]$
$constraints = [land(Where), minister(Who)]$