VEŠTAČKA INTELIGENCIJA

PLANIRANJE

Sadržaj

- Osnovne postavke
- Green-ova metoda
- STRIPS

Prof. dr Leonid Stoimenov

Planiranje

  - šta treba uraditi (**ciljevi**),
  - kako (i kada) to uraditi (**plan**).
- **Problem planiranja** - kako doći od trenutnog stanja do željenog ciljnog stanja?
- **Planiranje uključuje**
  - Selekciju akcija, Sekvencu akcija i Upravljanje resursima
- **Planovi mogu biti**
  - sekvence akcija
  - politika/strategije (stablo akcija)

Problemi u realnom svetu...

- Zato što je realni svet...
  - Dinamički
  - Stohastički
  - Delimično dostupan
- Zato što Akcije...
  - Zahtevaju vreme
  - Imaju kontinualne efekete

⇒ **Fokus na klasično planiranje**

⇒ Determinističko, statičko, potpuno dostupno

Rešavanje problema: planiranje ili traženje?

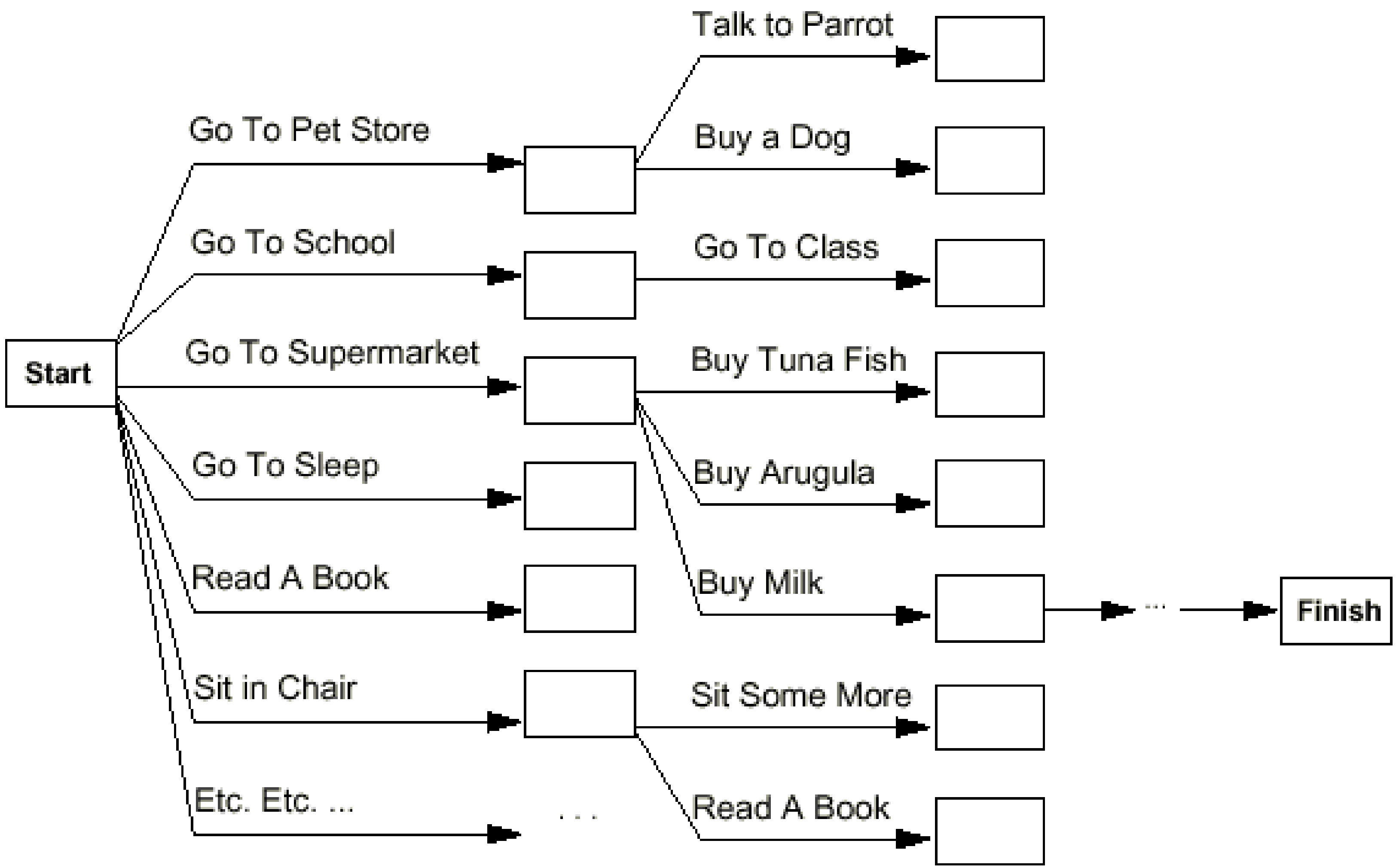
- **Planiranje** je vrsta/tehnika za rešavanje problema
  - zaključivanje o budućim događajima ...
  - ... sa ciljem da se verifikuje postojanje razumnih sekvenci akcija ...
  - ... da bi se postigao određeni cilj.
- Planiranje je **proces odlučivanja** u nizu akcija koji prethodi njihovom izvršenju.
- Agent koristi znanje o akcijama i njihovim posledicama da nadje rešenje u prostoru.

Rešavanje problema: planiranje ili traženje?

- **Planiranje kao traženje:**
  - traženje u prostoru stanja,
  - ... u kome su promene sa jednog na drugo stanje određene akcijama.
- Algoritmi za traženje
  - ne poseduju znanje o tome koje akcije su korisne u odredjenoj situaciji.
  - Ne mogu da odbace neke akcije kao nepotrebne u pojedinim koracima.

Planiranje kao traženje

Problem: kupi jabuke, hleb i cipele



Domen planiranja

- Fokus je na **reprezentacijama relacija**, preko **predikata** i **objekata**.
- Za svaki domen formira se specifičan first-order jezik koji sadrži predikate i funkcije korisne za opis domena.
- Domen planiranja je definisan preko **skupa operatora** koji su parametrizovana reprezentacija promena koje mogu da se dese u domenu.

Problem planiranja

Za domen planiranja D, standardni problem planiranja:

- **Ulazi**
- **Prostor stanja**
- **Izlaz**

Problem planiranja

- **Ulazi**
  - Početno stanje (sveta)
  - Cilj (koji treba dostići)
  - Domen D

Problem planiranja

- **Prostor stanja**
  - Sva stanja do kojih se može doći sekvencom akcija od početnog stanja
  - Akcije su operatori sa navedenim parametrima,
  - Akcije su instancirane na osnovu konstanti koje definišu početno stanje.

Problem planiranja

- **Izlaz**
  - **Plan**:

Sekvenca akcija koja transformiše incijalno stanje u stanje koje zadovoljava cilj.

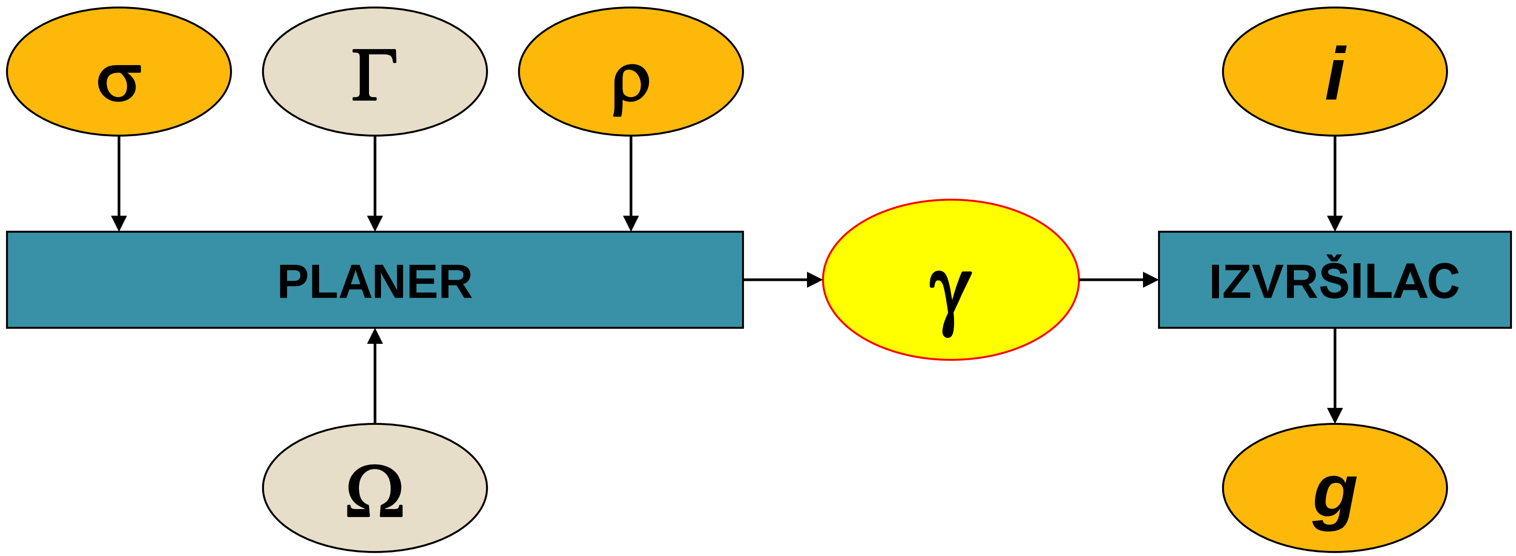
Agent koji planira ima **bazu znanja** koja opisuje akcije, opis cilja, i na osnovu opažanja formira plan koji kasnije izvršava.

Klasično planiranje

Preduslovi (za većinu klasičnih planera)

- Postoji kompletno znanje o incijalnom stanju.
- Cilj je konkretno individualno stanje sveta.
- Akcije menjaju svet iz jednog statičkog stanja u drugo.
- Akcije su determinističke
  - njihovi efekti su kompletno specificirani i predvidljivi,
  - ne modifikuju skup objekata u posmatranom svetu / domenu,
  - Promene u svetu se odnose jedino na rezultat primene akcija

Formulacija Planiranja / Green



Bazira se na poznavanju posledica mogućih akcija i upotrebi tog znanja radi postizanja nekog cilja.

Polazeći od opisa σ (polazno stanje), Γ (skup akcija), ρ (cilj), i Ω (baza formula o polaznom stanju, cilju i akcijama) odrediti **plan** γ.

Green-ova formulacija planiranja

- **Plan** mora da zadovolji uslove:
  - sve akcije koje se javljaju u planu moraju biti elementi skupa akcija Γ
  - mora postojati dokaz iz Ω da plan γ dovodi do stanja koje zadovoljava cilj ρ počevši od polaznog stanja σ
- Plan je akcija ili skup akcija koji su ulaz u IZVRŠILAC – primenom akcija na polazno stanje **i** daje stanje **g** koje zadovoljava opis cilja **i**.

Green-ova formulacija planiranja

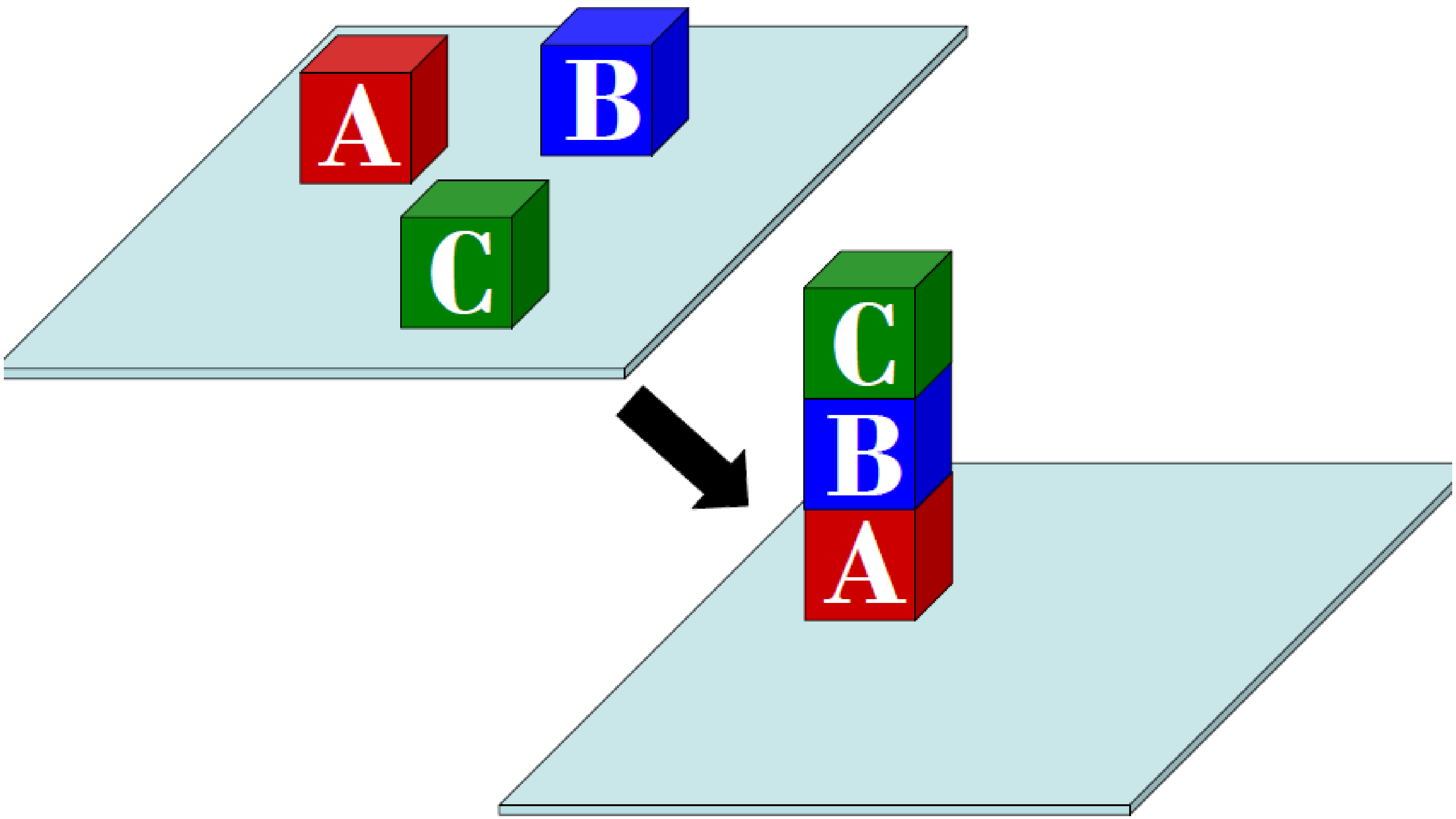
- Posmatra planiranje kao dokazivanje teorema
- Bazirana na **rezoluciji** (odgovaranje na pitanja)
- Polazi se od formule

$$\exists p\ Cilj(Result(p,s))$$

- Akcije se uredjuju u blokove (nizove akcija) $Result(l,s)$
  - $\forall s\ \mathrm{Result}([\ ],s) = s$
  - $\forall a\ \forall p\ \forall s\ \mathrm{Result}([a|p],s) = \mathrm{Result}(p, \mathrm{Result}(a,s))$

Planiranje u svetu blokova

- Sto i blokovi, ruka robota koja pomera blokove
- Dimenzije, oblik, ostali atributi ?



Stanja u svetu blokova

- **Stanje** je prikaz sveta u određenom trenutku vremena.
- Za svako stanje uvodi se **oznaka**, pa se stanja tretiraju kao neki objekti.
- Da bi se opisalo stanje uvodi se **binarni predikat**

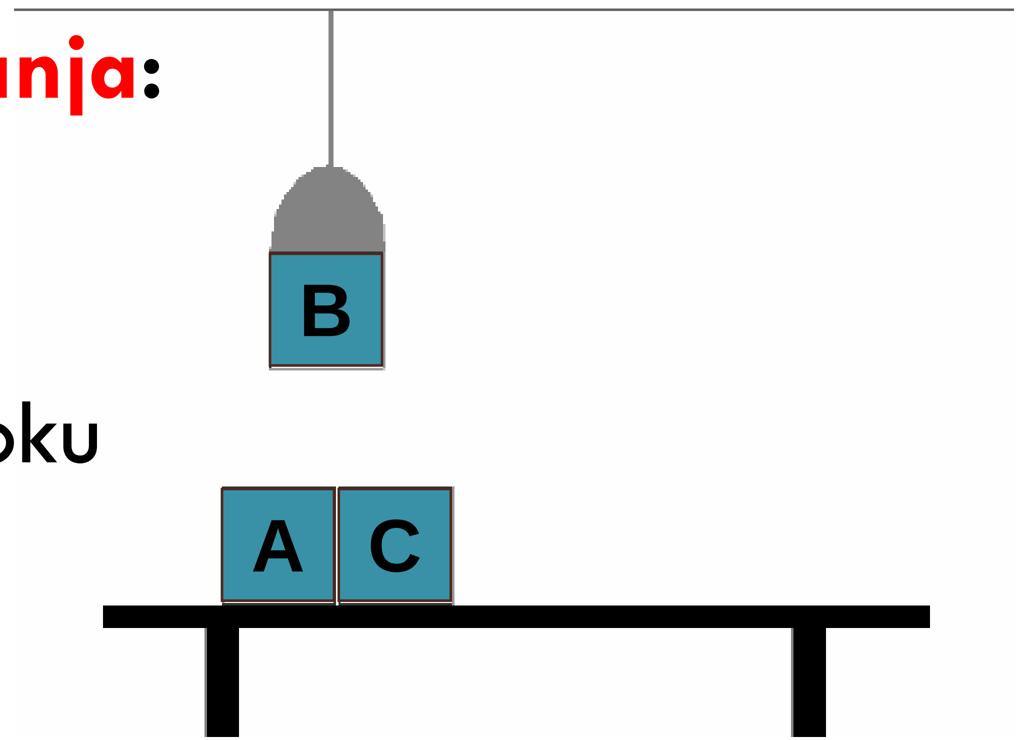
  T(*stanje_bloka, oznaka_stanja*)

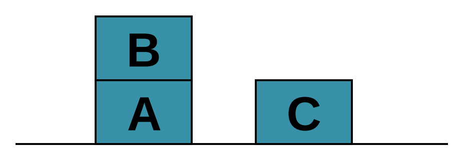
  koji kaže da je neka osobina tačna u određenom stanju.

Svet blokova

Predikati za opis stanja:

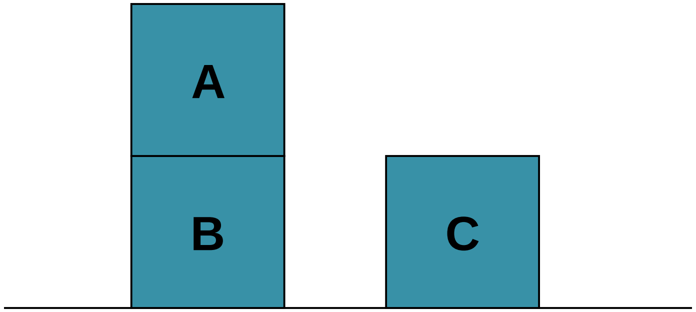
- Blok na stolu
  - Predikat ***Table(x)***
- Blok na drugom bloku
  - Predikat ***On(x,y)***
- Slobodan blok
  - Predikat ***Clear(x)***





Primer

- Stanje S2:



Opis stanja S2:

S2:

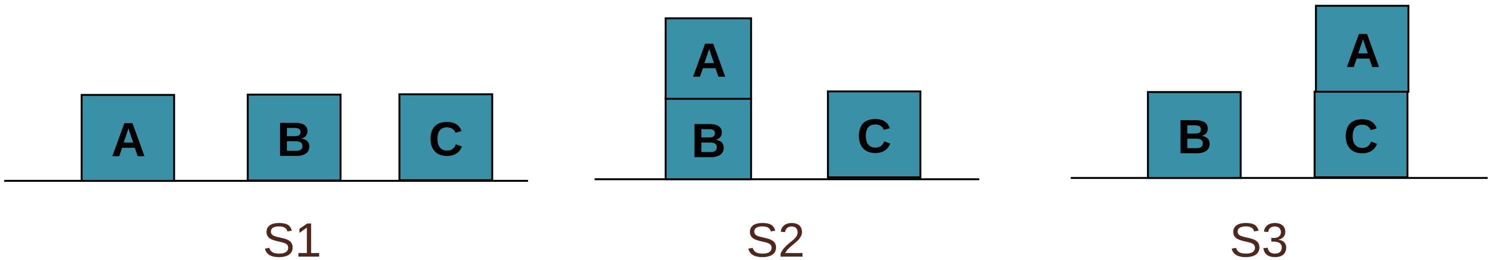
T(On(A, B), S2)
T(Table(B), S2)
T(Table(C), S2)
T(Clear(A), S2)
T(Clear(C), S2)

Akcije

- Operatori koji definišu promene u domenu
- Za **svet blokova** treba odrediti skup mogućih akcija u tom svetu preko operatora.
- **Operatori:**
  - **U – unstack, U(x, y)** - skida blok ***x*** sa bloka ***y*** i stavlja ga na sto
  - **S – stack, S(x, y)** - stavlja blok ***x*****,** koji se nalazi na stolu, na blok ***y***
  - **M – move, M(x, y, z)** - pomera blok ***x*****,** koji se nalazi na bloku ***y*****,** na blok ***z***
  - **Null** – not, NOOP

Funkcija ***Result***

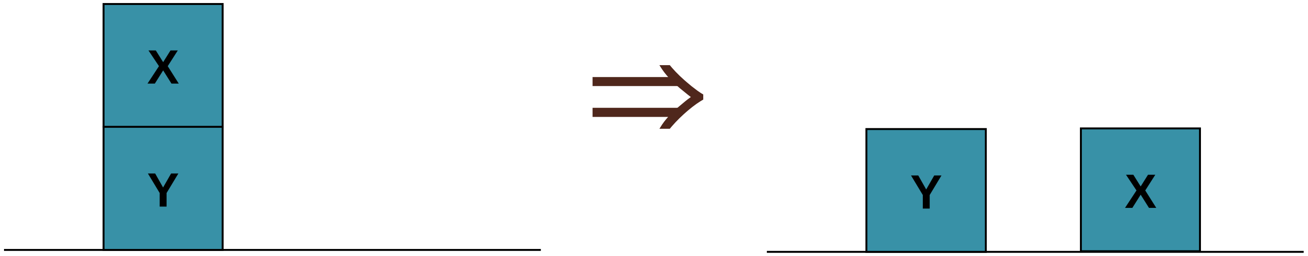
- Funkcija ***Result*** se koristi za definiciju novog stanja posle primene nekog od operatora:
  - **Result(*operator*, *stanje*) → *novo_stanje***
- Primeri:
  - **Result**(S(A, B), S1) → S2
  - **Result**(M(A,B,C), Result(S(A, B), S1)) → S3



Definicije operatora

Preduslovi za izvršenje operatora ⇒
Posledice izvršenja operatora

- Definicija operatora **U(x, y):**

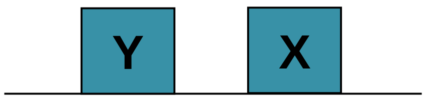


$$T(On(x, y), s) \wedge T(Clear(x), s) \Rightarrow T(Table(x), Result(U(x, y), s)) \wedge T(Clear(y), Result(U(x, y), s))$$

Definicije operatora

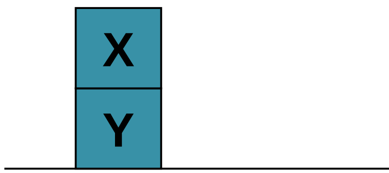
- Definicija operatora **S(x, y)**:

$T(Table(x), s) \wedge T(Clear(y), s)$



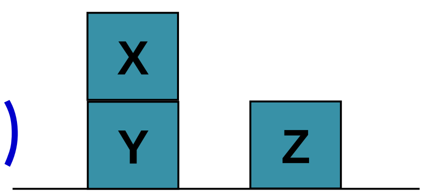
$\Rightarrow$

$T(On(x, y), Result(S(x, y), s))$



- Definicija operatora **M(x, y, z)**:

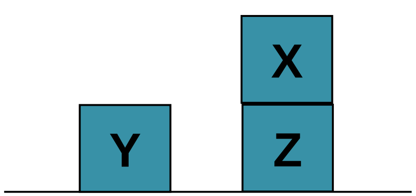
$T(Clear(x), s) \wedge T(On(x, y), s) \wedge T(Clear(z), s)$



$\Rightarrow$

$T(On(x, z), Result(M(x, y, z), s)) \wedge$

$T(Clear(y), Result(M(x, y, z), s))$



Aksiome okvira

- *Frame* aksiome
- Problem okvira: bitno je znati i šta se nije promenilo u svetu.
- Koriste se da definišu prelaze iz tekućeg stanja u novo stanje blokova koji nisu pod uticajem operatora.
- **Za svaki od operatora U, S i M definiše se po jedna *frame* aksioma za stanja blokova Clear(x), Table(x), On(x, y).**

Aksiome okvira za *Unstack*

- Izvršena je akcija **U(x, y)** nad stanjem **s**:
  - $T(Clear(u), s) \Rightarrow T(Clear(u), Result(U(x, y), s))$
  - $T(Table(u), s) \Rightarrow T(Table(u), Result(U(x, y), s))$
  - $T(On(u, w), s) \wedge u \neq x \Rightarrow$
    $T(On(u, w), Result(U(x, y), s))$

Aksiome okvira za Stack

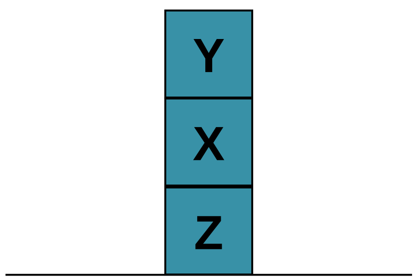
- Izvršena je akcija **S(x, y)** nad stanjem **s:**
  - $T(Clear(u), s) \wedge u \neq y \Rightarrow$
    $T(Clear(u), Result(S(x, y), s))$
  - $T(Table(u), s) \wedge u \neq x \Rightarrow$
    $T(Table(u), Result(S(x, y), s))$
  - $T(On(u, w), s) \Rightarrow$
    $T(On(u, w), Result(S(x, y), s))$

Aksiome okvira za Move

- Izvršena je akcija **M(x, y, z)** nad stanjem **s:**
  - $T(Clear(u), s) \wedge u \neq z \Rightarrow$
    $T(Clear(u), Result(M(x, y, z), s))$
  - $T(Table(u), s) \Rightarrow$
    $T(Table(u), Result(M(x, y, z), s))$
  - $T(On(u, w), s) \wedge u \neq x \Rightarrow$
    $T(On(u, w), Result(M(x, y, z), s))$

Cilj

- **Cilj** je bilo koje željeno stanje u svetu blokova.
- Primer ciljnog stanja ***t***



- Opis cilja:

  $T(On(A, B), t) \wedge T(On(B, C), t) \Leftrightarrow Goal(t)$

Planiranje u svetu blokova

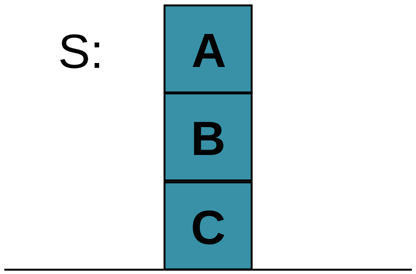
- Planiranje se zasniva na predikatskoj logici:
  - Sve akcije moraju biti iz skupa {U, S, M}
  - Mora postojati dokaz da skup akcija prevodi početno stanje u ciljno stanje.
  - Izvođenje plana se zasniva na rezoluciji.

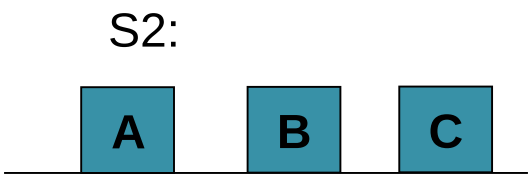
Blok akcija

- Plan može da se sastoji iz **bloka akcija**
  - Blok akcija je konačan niz akcija.
  - Primenjuju se redom
  - [ ] **–** oznaka za prazan blok akcija
  - Primer

    Result([ ], S) = S

    Result([U(A, B), U(B, C)], S) = S2





Plan

- Plan ***a*** koji se sastoji iz skupa akcija po Green-ovoj metodi se dobija primenom rezolucije
- Teorema koja se dokazuje:

  Goal(Result(a, S1)) $\Rightarrow$ Ans(a)
- Predikat **Goal** definiše ciljno stanje,
- Predikat **Ans** definiše plan koji prevodi početno stanje u ciljno.

  Goal(t) $\Leftrightarrow$ ***opis ciljnog stanja t***

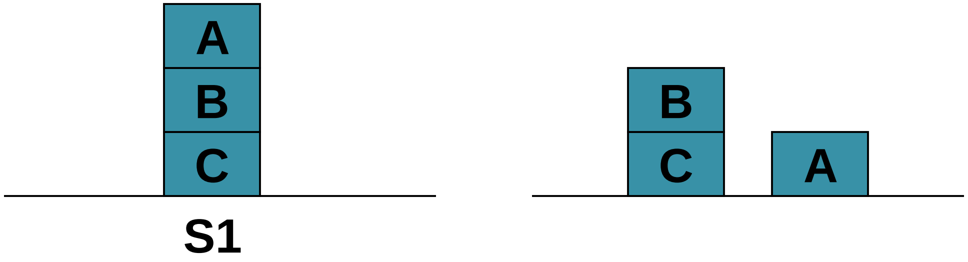
Ulazni podaci za Green-ovu metodu

- **Polazno stanje**
  - Opis polaznog stanja
- **Ciljno stanje** (teorema koju treba dokazati)
  - Opis ciljnog stanja
- **Definicija operatora**
- ***Frame* aksiome**

- Primena **rezolucije** za odgovaranje na pitanja
  - Negacija teoreme
  - Dodatni literal

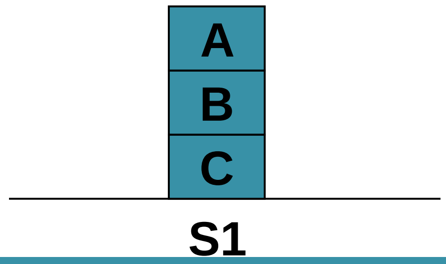
Primer 1 – plan od jedne akcije

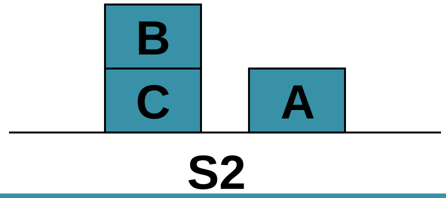
- Izvesti plan za prevođenje sveta blokova iz stanja S1 u stanje u kome je blok A na stolu.
- Rešenje:



  - Analizom problema se uočava da akcija U(A, B) prevodi svet blokova iz stanja S1 u ciljno stanje.

Primer 1 (nast.)



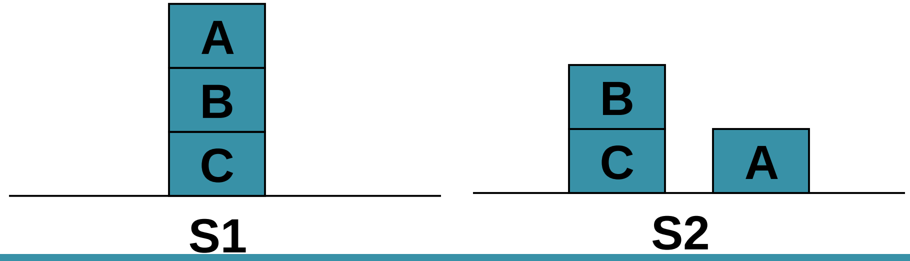


- Definicija početnog stanja S1 i odg. klauzule:

$T(Clear(A), S1) \wedge T(Table(C), S1) \wedge$
$T(On(A, B), S1) \wedge T(On(B, C), S1)$

$\{T(Clear(A), S1)\}$
$\{T(Table(C), S1)\}$
$\{T(On(A, B), S1)\}$
$\{T(On(B, C), S1)\}$

Primer 1 (nast.)



- Definicija cilja:

  Goal(t) $\Leftrightarrow$ T(Table(A), t)

- Teorema koju treba dokazati i odg. klauzula (nakon negacije):
  - T(Table(A), Result(a, S1)) $\Rightarrow$ Ans(a)
  - **{$\neg$T(Table(A), Result(a, S1)), Ans(a)}**

Primer 1 (nast.) – def. operatora

Operator U(x, y):

$$T(On(x, y), s) \wedge T(Clear(x), s) \Rightarrow T(Table(x), Result(U(x, y), s)) \wedge T(Clear(y), Result(U(x, y), s))$$

Klauzule:

$$\{\neg T(On(x, y), s), \neg T(Clear(x), s), T(Table(x), Result(U(x, y), s))\}$$

$$\{\neg T(On(x, y), s), \neg Clear(x), s), T(Clear(y), Result(U(x, y), s))\}$$

Primer 1 (nast.) – skup klauzula

1: {T(Clear(A), S1)}

2: {T(Table(C), S1)}

3: {T(On(A, B), S1)}

4: {T(On(B, C), S1)}

5: {T(Table(x), Result(U(x, y), s)),
¬T(On(x, y), s), ¬T(Clear(x), s)}

6: {T(Clear(y), Result(U(x, y), s)),
¬T(On(x, y), s), ¬Clear(x), s)}

7: {¬T(Table(A), Result(a, S1)), Ans(a)}

Primer 1 (nast.) - rezolucija

8. Poklapanjem klauzula 5 i 7

5: {T(Table(x), Result(U(x, y), s)), ¬T(On(x, y), s), ¬T(Clear(x), s)}

7: {¬T(Table(A), Result(a, S1)), Ans(a)}

sa listom smena:

(x / A; a / U(A, y); s / S1) dobijamo:

{Ans(U(A, y)), ¬T(On(A, y), S1), ¬T(Clear(A), S1)}

Ovim se tvrdi da ako postoji blok y takav da je A na y i A je slobodno, onda izvođenje U(A, y) vodi ka cilju.

Primer 1 (nast.) - rezolucija

9. Poklapanjem klauzula 3 i 8:

3: {T(On(A, B), S1)}

8: {Ans(U(A, y)), ¬T(On(A, y), S1), ¬T(Clear(A), S1)}

sa listom smena: (y / B) dobijamo

{Ans(U(A, B)), ¬T(Clear(A), S1)}

10. Poklapanjem klauzula 1: {T(Clear(A), S1)}

i 9 dobijamo:

{Ans(U(A, B))}

Primer 1 – kompletno rešenje

1: {T(Clear(A), S1)}
2: {T(Table(C), S1)}
3: {T(On(A, B), S1)}
4: {T(On(B, C), S1)}

Polazno stanje

5: {T(Table(x), Result(U(x, y), s)),
¬T(On(x, y), s), ¬T(Clear(x), s)}
6: {T(Clear(y), Result(U(x, y), s)),
¬T(On(x, y), s), ¬Clear(x), s)}

Operator

7: {¬T(Table(A), Result(a, S1)), Ans(a)}

Cilj (negacija)

8. {Ans(U(A, y)), ¬T(On(A, y), S1), ¬T(Clear(A), S1)}, iz 5 i 7, lista smena (x / A; a / U(A, y); s / S1)

Iz def.operatora i opisa cilja

9. {Ans(U(A, B)), ¬T(Clear(A), S1)} iz 3 i 8, lista smena (y / B)

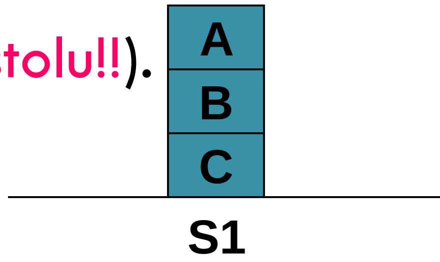
Iz opisa polaznog stanja

10. {Ans(U(A, B))}, iz 1 i 9

Primer 2: korišćenje aksioma okvira

- **Početno stanje:** kao kod prethodnog primera
- **Cilj:** stanje u kome su **A** i **C** na stolu (C je već na stolu!!).

  T(Table(A), t) $\wedge$ T(Table(C), t) $\Leftrightarrow$ Goal(t)



Dokaz:

- … (kao u prethodnom primeru, na osnovu def.operatora U), dobija se:

{¬T(On(A, y), S1), ¬T(Clear(A), S1), ¬T(Table(C), Result(U(A, y), S1)), Ans(U(A, y)) }

Poklapanjem sa klauzulama iz opisa polaznog stanja mogu da se eliminišu svi literali osim

¬T(Table(C), Result(U(A, y), S1))

1: {T(Clear(A), S1)}
2: {T(Table(C), S1)}
3: {T(On(A, B), S1)}
4: {T(On(B, C), S1)}

Primer: korišćenje aksioma okvira

{¬T(On(A, y), S1), ¬T(Clear(A), S1), ¬T(Table(C), Result(U(A, y), S1)), Ans(U(A, y)) }

1: {T(Clear(A), S1)}
2: {T(Table(C), S1)}
3: {T(On(A, B), S1)}
4: {T(On(B, C), S1)}

- **Frame axiom na U akciju:**

  T(Table(u), s) => T(Table(u), Result(U(x, y), s))

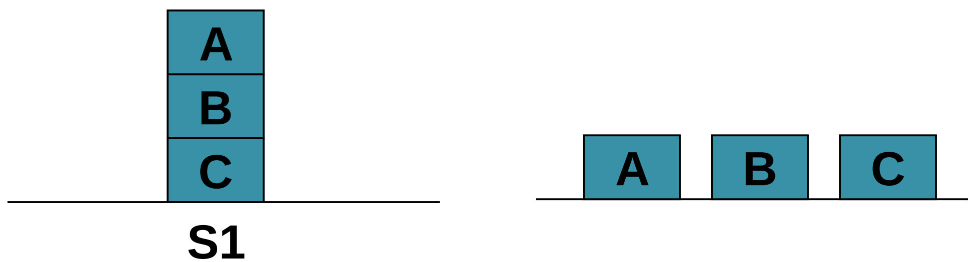
  Odgovarajuća klauzula

  {¬T(Table(u), s), T(Table(u), Result(U(x, y), s))}
- **Rezolucija:**

  {¬ T(On(A, y), S1), ¬ T(Clear(A), S1), ¬ T(Table(C), S1), Ans(U(A, y)) }
- Ostalo se dobija rezolucijom sa opisom polaznog stanja:
  - { **T(On(A, y), S1)**, T(Clear(A), S1), Ans(U(A, y)) }
  - { **T(Clear(A), S1)**, Ans(U(A, **B**)) }
  - **{Ans(U(A, B)) }**

Složeniji primer (blok akcija)

- Izvesti plan za prevođenje sveta blokova iz stanja S1 u stanje u kome je blok B na stolu.
- Rešenje:



  - Analizom problema se uočava da **blok akcija** U(A, B) i U(B, C) prevodi svet blokova iz stanja S1 ciljno stanje.

Složeniji primer (nast.)

- Definicija početnog stanja S1:
  - T(Clear(A), S1) $\wedge$
    T(Table(C), S1) $\wedge$
    T(On(A, B), S1) $\wedge$
    T(On(B, C), S1)
- Definicija cilja:
  - Goal(t) $\Leftrightarrow$ T(Table(B), t)
  - **Blok akcija**

    T(Table(B), Result(a2, Result(a1, S1))) $\Rightarrow$ Ans(a1, a2)

Složeniji primer (nast.) - klauzule

- **Početno stanje:**

  P1: {T(Clear(A), S1)}

  P2: {T(Table(C), S1)}

  P3: {T(On(A, B), S1)}

  P4: {T(On(B, C), S1)}

- **Operator U(x, y):**

  O1: {T(Table(x), do(U(x, y), s)),

  ¬T(On(x, y), s), ¬T(Clear(x), s)}

  O2: {T(Clear(y), do(U(x, y), s)),

  ¬T(On(x, y), s), ¬Clear(x), s)}

Složeniji primer (nast.)

- Definicija **cilja** (<u>blok od dve akcije</u>):

  C: {¬T(Table(B), Result(a2, Result(a1, S1))),

  Ans(a1, a2)}

- **Frame aksiome** za operator **U**:

$$T(On(u, w), s) \wedge u \neq x \;\Rightarrow\; T(On(u, w), Result(U(x, y), s))$$

F: {¬T(On(u, w), s), ¬u≠x,

T(On(u, w), do(U(x, y), s))}

Složeniji primer (nast.) - rezolucija

- Koristi se opis operatora U
- Poklapanjem klauzula **C(ilj)** i **O1** (iz def.operatora), sa listom smena:

  (x / B; a2 / U(B, y); s / Result(a1, S1))

  dobijamo:

  R1: {Ans(a1, U(B, y)),
  ¬T(On(B, y), Result(a1, S1)),
  ¬T(Clear(B), Result(a1, S1))}

Složeniji primer (nast.) - rezolucija

- Ponovo se koristi opis operatora U (druga klauzula)
- Poklapanjem klauzula R1 i O2, sa listom smena: (y / B; a1 / U(x, B); s / S1)
  - NAPOMENA: *y* je promenljiva iz klauzule O2 i nema veze sa promenljivom iz klauzule R1 (korak 8 – preimenovanje promenljivih!!).

dobijamo:

R2: {Ans(U(x, B), U(B, y)),

$\neg$T(On(B, y), Result(U(x, B), S1)),

$\neg$T(On(x, B), S1), $\neg$T(Clear(x), S1)}

Složeniji primer (nast.) - rezolucija

Poklapanje sa klauzulama iz opisa polaznog stanja:

R2: {Ans(U(x, B), U(B, y)),

¬T(On(B, y), Result(U(x, B), S1)),

¬T(On(x, B), S1), ¬T(Clear(x), S1)}

P1: {T(Clear(A), S1)}
P2: {T(Table(C), S1)}
P3: {T(On(A, B), S1)}
P4: {T(On(B, C), S1)}

Složeniji primer (nast.) - rezolucija

- Poklapanjem klauzula R2 i P1, sa listom smena: (x / A) dobijamo:

  R3: {Ans(U(A, B), U(B, y)),

  ¬T(On(B, y), do(U(A, B), S1)),

  ¬T(On(A, B), S1)}

- Poklapanjem klauzula R3 i P3 dobijamo:

  R4: {Ans(U(A, B), U(B, y)),

  ¬T(On(B, y), do(U(A, B), S1))}

Složeniji primer (nast.) - rezolucija

- Poklapanjem klauzule R4 sa frame aksiomom F sa listom smena:
  (u / B; x / A, y / B, s / S1, w / y) dobijamo:
  R5: {Ans(U(A, B), U(B, y)),
  ¬T(On(B, y), S1), ¬B≠A}
- Kako su A i B različiti blokovi možemo da dodamo podrazumevanu klauzulu: **{B≠A}**
- Poklapanjem klauzule R5 sa podrazumevanom klauzulom dobijamo:
  R6: {Ans(U(A, B), U(B, y)), ¬T(On(B, y), S1)}

Složeniji primer (nast.) - rezolucija

- Konačno, poklapanjem klauzula R6 i P4, sa listom smena: (y / C) dobijamo odgovor:

R7: {Ans(U(A, B), U(B, C))}

- Redosled akcija koje treba sprovesti da bi se svet blokova preveo iz stanja S1 u stanje S2 je:

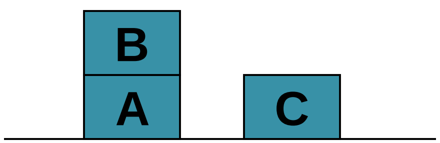
1. **U(A, B),**
2. **U(B, C).**

STRIPS algoritam

- Cilj STRIPS-a je da se nađe sekvenca operatora koja sistem prevodi iz početnog stanja u ciljno stanje.
- STRIPS algoritam koristi sledeće strukture podataka:
  - **Tekuće stanje** problema opisano u predikatskoj logici.
    - Inicijalno je to opis početnog stanja.
  - **Ciljni stek** koji sadrži stavove koji odgovaraju trenutnom (pod)cilju.
    - Inicijalno je to opis ciljnog stanje.
  - **Lista akcija** koja na kraju sadrži sekvencu operacija koja predstavlja plan
    - Inicijalno je lista prazna.

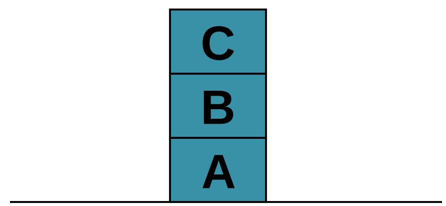
Opis stanja

- Za opis stanja i cilja koriste se ranije definisani predikati:
  - ***Clear(x)*** - nijedan blok ne stoji na bloku ***x***
  - ***Table(x)*** - blok ***x*** se nalazi na tabli
  - ***On(x, y)*** - blok ***x*** se nalazi na bloku ***y***



S1

Clear(B)
Table(A)
Table(C)
On(B, A)



S2

Clear(C)
Table(A)
On(C, B)
On(B, A)

S3

Clear(A) Table(A)
Clear(B) Table(B)
Clear(C) Table(C)

Operatori

- **Koriste se ranije definisani operatori u svetu blokova**
  - **U(x, y)** - *unstack*
    - skida blok **x** sa bloka ***y*** i stavlja ga na tablu
  - **S(x, y)** - *stack*
    - stavlja blok **x**, koji se nalazi na tabli, na blok ***y***
  - **M(x, y, z)** - *move*
    - pomera blok **x**, koji se nalazi na bloku ***y***, na blok ***z***

Operatori promene stanja

- Operatori se definišu zadavanjem tri liste stavova:
  - **PREDUSLOV** - lista stavova koja moraju biti ispunjeni u tekućem stanju da bi operator mogao biti primenjen.
  - **UKLONI** - lista stavova koji se uklanjanju iz tekućeg stanja u trenutku primene operatora.
  - **DODAJ** - lista stavova koji se dodaju tekućem stanju nakon primene operatora.

Definicije operatora

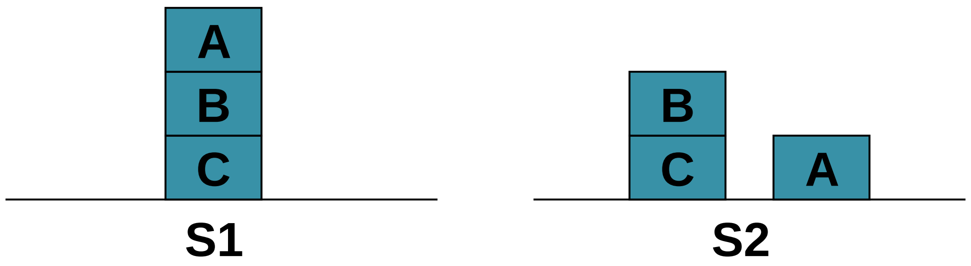
- ***U(x, y)***
  - PREDUSLOV: ***Clear(x), On(x, y)***
  - UKLONI: ***On(x, y)***
  - DODAJ: ***Table(x), Clear(y)***
- ***S(x, y)***
  - PREDUSLOV: ***Table (x), Clear(y)***
  - UKLONI: ***Table(x), Clear(y)***
  - DODAJ: ***On(x, y)***
- ***M(x, y, z)***
  - PREDUSLOV: ***Clear(x), On(x, y), Clear(z)***
  - UKLONI: ***On(x, y), Clear(z)***
  - DODAJ: ***On(x, z), Clear(y)***

Opis algoritma

- Definiše se početno stanje.
- Definiše se cilj i postavi na stek.
- Dok stek nije prazan uzima se stav sa vrha steka:
  - Ako je stav (pod)cilj proveriti da li je zadovoljen:
    - Ako je zadovoljen uklanja se sa steka.
    - U suprotnom se bira operator koji ga zadovoljava i stavlja na stek. Stavljaju se i svi njegovi PREDUSLOVI.
  - Ako je stav operator:
    - Skida se sa steka i dodaje se listi akcija.
    - Uklanjaju se stavovi prema listi UKLONI iz tekućeg stanja.
    - Dodaju se stavovi prema listi DODAJ u tekuće stanje.
- Izdati listu akcija.

Zadatak 1.

- Koristeći STRIPS algoritam naći sekvencu operatora koji prevode svet blokova iz stanja S1 u stanje S2:



Zadatak 1. (rešenje)

- Inicijalno stanje STRIPS struktura je:
  - Tekuće stanje:
    - Clear(A), Table(C), On(A, B), On(B, C)
  - Ciljni stek:
    - Table(A)
  - Lista akcija je inicijalno prazna.
- Uzimamo stav sa vrha steka -> **Table(A)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto ga nema u tekućem stanju biramo operator koji ga ima u DODAJ delu -> **U(x, y)**
  - Izvrši se unifikacija (pod)cilja i DODAJ dela operatora **U(x, y)**, a zatim (ukoliko je potrebno) i PREDUSLOVA operatora sa tekućim stanjem:
    - Dobija se lista smena: (**x** / **A**, **y** / **B**)

Zadatak 1. (rešenje)

- Novo stanje ciljnog steka sadrži operator i njegove preduslove (prvo se dodaje operator, pa onda preduslovi):
  - Clear(A)
  - On(A, B)
  - U(A, B)
  - Table(A)
- Uzimamo stav sa vrha steka -> **Clear(A)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto se (pod)cilj nalazi u tekućem stanju uklanja se sa steka.
- Uzimamo stav sa vrha steka -> **On(A, B)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto se (pod)cilj nalazi u tekućem stanju uklanja se sa steka.

Zadatak 1. (rešenje)

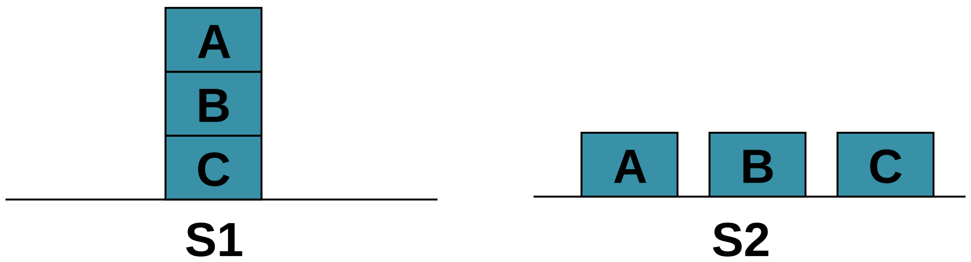
- Tekuće stanje:
  - Clear(A), Table(C), On(A, B), On(B, C)
- Ciljni stek:
  - U(A, B)
  - Table(A)
- Uzimamo stav sa vrha steka -> **U(A, B)**
  - Kako je reč o operatoru, skida se sa steka i dodaje se **listi akcija**.
  - Stavovi sadržani u UKLONI listi operatora U(A, B) se uklanjaju iz tekućeg stanja.
  - Stavovi sadržani u DODAJ listi operatora U(A, B) se dodaju u tekuće stanje.

Zadatak 1. (rešenje)

- Tekuće stanje:
  - Clear(A), Table(C), On(B, C), Table(A), Clear(B)
- Ciljni stek:
  - Table(A)
- Uzimamo stav sa vrha steka -> **Table(A)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto ga ima u tekućem stanju, uklanja se iz steka.
- Stek je nakon ovog koraka prazan.
  - Izdaje se lista akcija: **U(A, B)**

Zadatak 2.

- Koristeći STRIPS algoritam naći sekvencu operatora koji prevode svet blokova iz stanja S1 u stanje S2:



Zadatak 2. (rešenje)

- Inicijalno stanje STRIPS struktura je:
  - Tekuće stanje:
    - Clear(A), Table(C), On(A, B), On(B, C)
  - Ciljni stek:
    - Table(B)
  - Lista akcija je inicijalno prazna.
- Uzimamo stav sa vrha steka -> **Table(B)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto ga nema u tekućem stanju biramo operator koji ga ima u DODAJ delu -> **U(x, y)**
  - Izvrši se unifikacija (pod)cilja i DODAJ dela operatora **U(x, y)**, a zatim (ukoliko je potrebno) i PREDUSLOVA operatora sa tekućim stanjem:
    - Dobija se lista smena: **(x** / **B**, **y** / **C**)

Zadatak 2. (rešenje)

- Novo stanje ciljnog steka sadrži operator i njegove preduslove (prvo se dodaje operator, pa onda preduslovi):
  - Clear(B)
  - On(B, C)
  - U(B, C)
  - Table(B)
- Uzimamo stav sa vrha steka -> **Clear(B)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto ga nema u tekućem stanju biramo operator koji ga ima u DODAJ delu -> **U(x, y)** ili **M(x, y, z)**
  - Koristeći heuristiku bira se operator **U(x, y)** jer je moguća unifikacija sa (pod)ciljem (y / B) i preduslova sa tekućim stanjem (x / A).

Zadatak 2. (rešenje)

- Novo stanje ciljnog steka sadrži operator i njegove preduslove (prvo se dodaje operator, pa onda preduslovi):
  - Clear(A)
  - On(A, B)
  - U(A, B)
  - Clear(B)
  - On(B, C)
  - U(B, C)
  - Table(B)
- Uzimamo stav sa vrha steka -> **Clear(A)**
  - Pošto se (pod)cilj nalazi u tekućem stanju uklanja se sa steka.
- Uzimamo stav sa vrha steka -> **On(A, B)**
  - Pošto se (pod)cilj nalazi u tekućem stanju uklanja se sa steka.

Zadatak 2. (rešenje)

- Tekuće stanje:
  - Clear(A), Table(C), On(A, B), On(B, C)
- Ciljni stek:
  - U(A, B)
  - Clear(B)
  - On(B, C)
  - U(B, C)
  - Table(B)
- Uzimamo stav sa vrha steka -> **U(A, B)**
  - Kako je reč o operatoru, skida se sa steka i dodaje se **listi akcija**.
  - Stavovi sadržani u UKLONI listi operatora U(A, B) se uklanjaju iz tekućeg stanja.
  - Stavovi sadržani u DODAJ listi operatora U(A, B) se dodaju u tekuće stanje.

Zadatak 2. (rešenje)

- Tekuće stanje:
  - Clear(A), Table(C), On(B, C), Table(A), Clear(B)
- Ciljni stek:
  - Clear(B)
  - On(B, C)
  - U(B, C)
  - Table(B)
- Lista akcija: U(A, B)
- Uzimamo stav sa vrha steka -> **Clear(B)**
  - Pošto ga ima u tekućem stanju, uklanja se iz steka.
- Uzimamo stav sa vrha steka -> **On(B, C)**
  - Pošto ga ima u tekućem stanju, uklanja se iz steka.

Zadatak 2. (rešenje)

- Tekuće stanje:
  - Clear(A), Table(C), On(B, C), Table(A), Clear(B)
- Ciljni stek:
  - U(B, C)
  - Table(B)
- Lista akcija: U(A, B)
- Uzimamo stav sa vrha steka -> **U(B, C)**
  - Kako je reč o operatoru, skida se sa steka i dodaje se **listi akcija**.
  - Stavovi sadržani u UKLONI listi operatora U(B, C) se uklanjaju iz tekućeg stanja.
  - Stavovi sadržani u DODAJ listi operatora U(B, C) se dodaju u tekuće stanje.

Zadatak 2. (rešenje)

- Tekuće stanje:
  - Clear(A), Table(C), Table(A), Clear(B), Table(B), Clear(C)
- Ciljni stek:
  - Table(B)
- Lista akcija: U(A, B), U(B, C)
- Uzimamo stav sa vrha steka -> **Table(B)**
  - Kako je reč o (pod)cilju proveravamo da li je zadovoljen.
  - Pošto ga ima u tekućem stanju, uklanja se iz steka.
- Stek je nakon ovog koraka prazan.
  - Izdaje se lista akcija: **U(A, B), U(B, C)**

PITANJA?

Dileme?

Komentari?