Cselekvési tervek generálása a robotikában

Nagy Tímea, Régeni Ágnes

Tartalom

- Robotika bevezető
  - Meghatározás
  - Osztályozás
  - Jellemzők
  - Robotgenerációk
- Cselekvési tervek
  - Bevezető
  - Algoritmusok
  - Példák (PROLOG)

Robotika

- A megnevezés Karel Capek színdarabjából (1921), a cseh ´ROBOTA´ szóból származik, amely munkát jelent.
- Általában olyan eszközt, berendezést értünk rajta, amely az ember fizikai és/vagy szellemi munkájához hasonló tevékenységet végez.
- Rendelkezniük kell számos, sok esetben az intelligencia elemei közé tartozó, tulajdonsággal (tudás, emlékezet, tanulási képesség, döntéseken alapuló közlési-cselekvési képesség stb.).

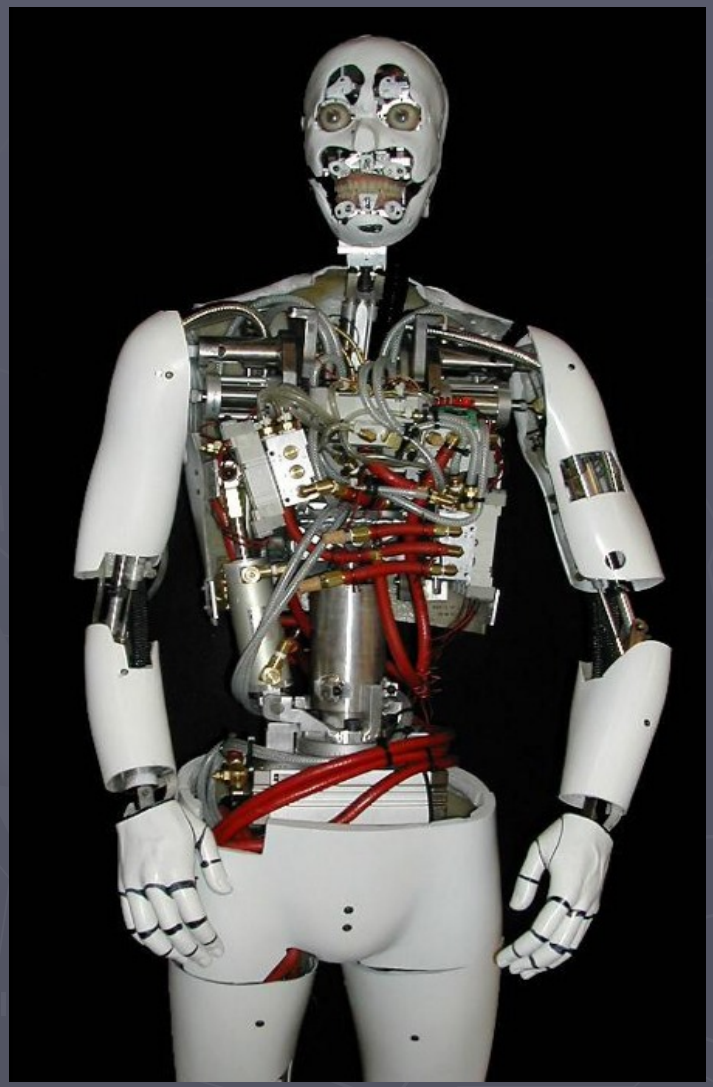
Jellemzői

- Teljes egészében ember által készített szerkezetek.
- A robotok egyik legfontosabb tulajdonsága a helyváltoztatási képesség, illetve annak hiánya.
- Tevékenységüket részben vagy teljesen önállóan irányítják.

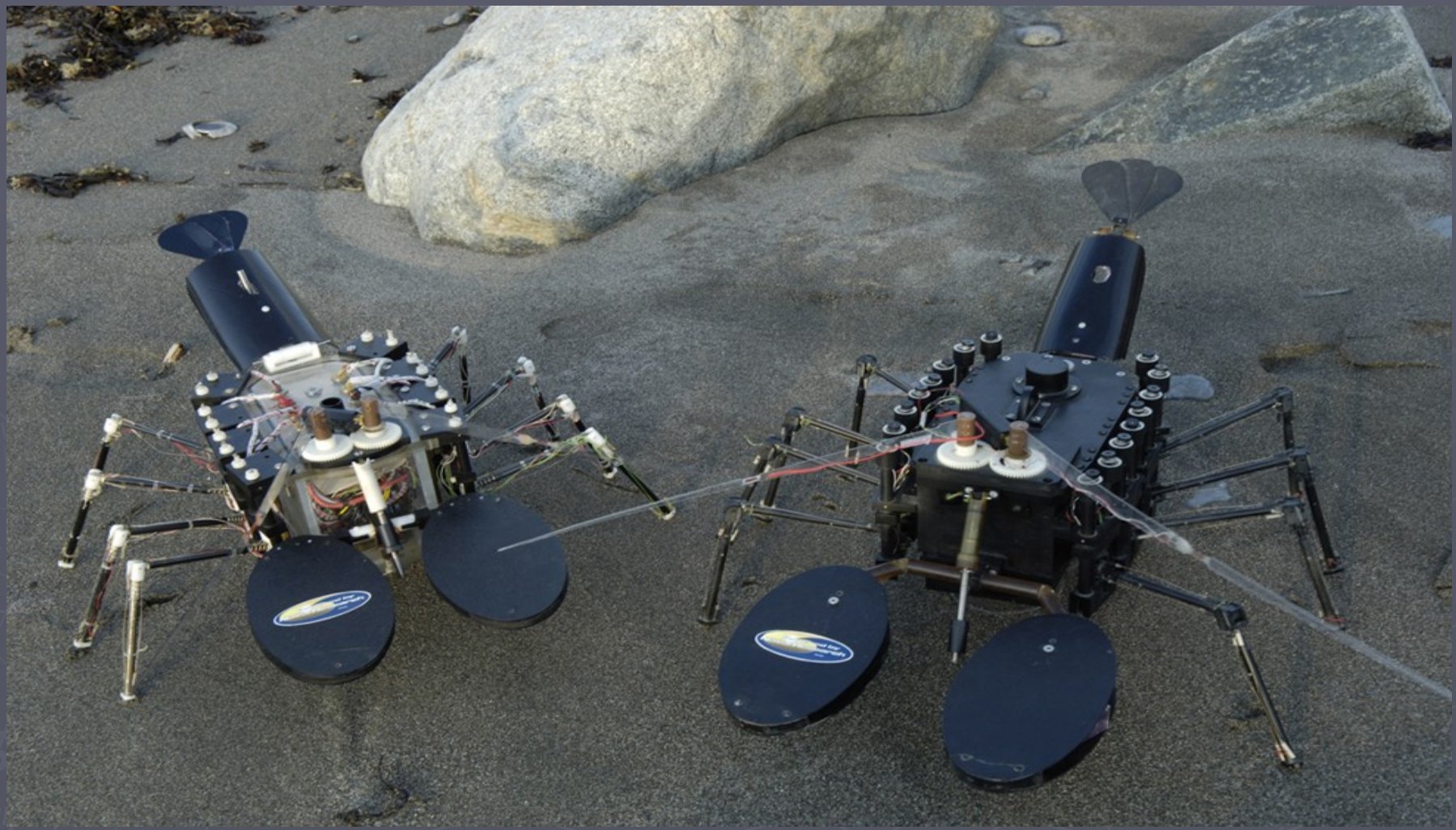
Osztályozás

- Megkülönböztetünk **mobil, statikus,** illetve **speciális** robotokat.
- Mobil robotok: **androidok**, **animatok**, **ember nélküli járművek**, **szórakoztató robotok** és az általános **autonóm** robotok.
- Statikus robotok: a **háztartási és ipari robotok** elsősorban a robotkarok, de ez nem követelmény.
- Speciális robotok a **nanorobotok**, a fizikai és kémiai területeken fejlesztik.

Androidok



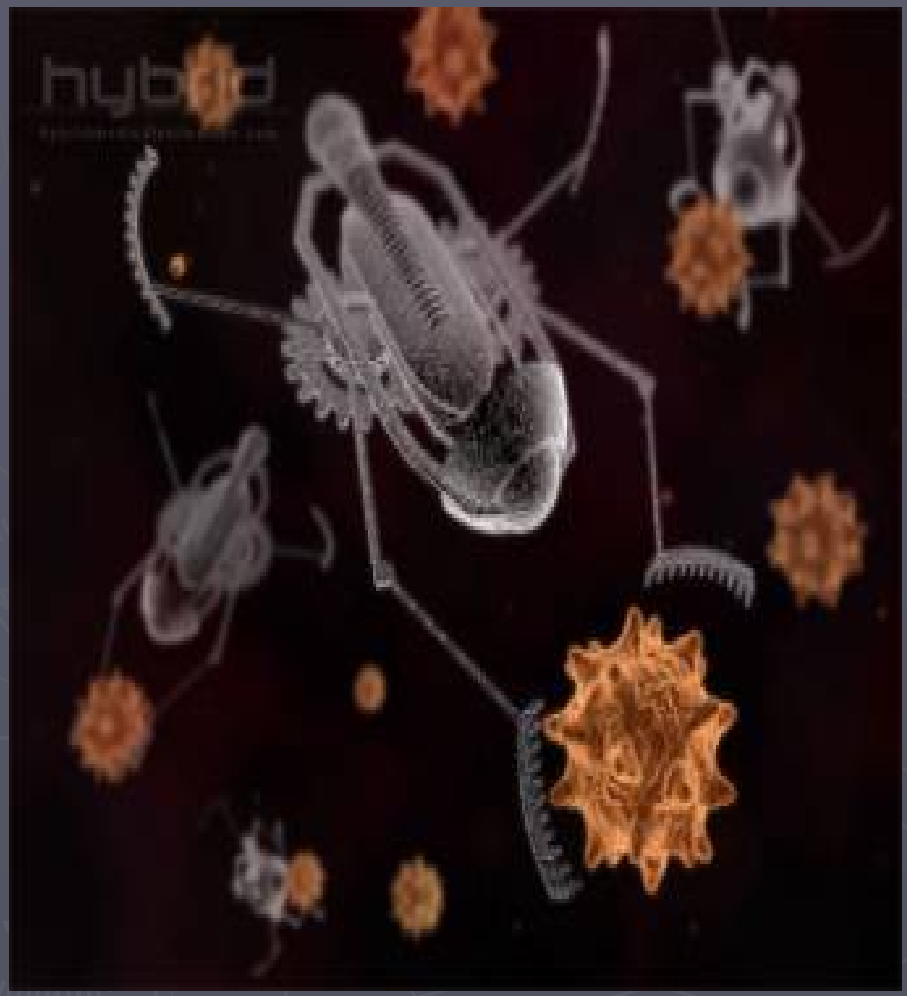
Animatok



Szórakoztató robotok

Ipari robotok

Nanorobotok



Robotgenerációk

- **Első** generációs robotok:
  - Kizárólag kézi vezérléssel működtethetőek;
  - nem érzékelik a környezet változásait;
  - a számítógép programja kap főszerepet.

Robotgenerációk

- **Második** generációs robotok:
  - környezetüket szenzorokkal vizsgálják;
  - képesek kikerülni az útjukba kerülő akadályokat;
  - a feladataikat magas szintű programozási nyelven határozzák meg.

Robotgenerációk

- **Harmadik** generációs robotok:
  - jól alkalmazkodnak a környezet változásaihoz;
  - alakokat és helyeket ismernek fel;
  - hanggal is vezérelhetők és képesek hanggal válaszolni;
  - önálló döntéseket is hoznak;
  - bonyolult feladatokat oldanak meg;
  - tanuló algoritmusokat használnak.

Cselekvési tervek

- szervezetek, csoportok vagy egyének bizonyos célok elérésére irányuló viselkedésének meghatározása
- az elérendő célok a cselekvőt körülvevő világhoz kötődnek

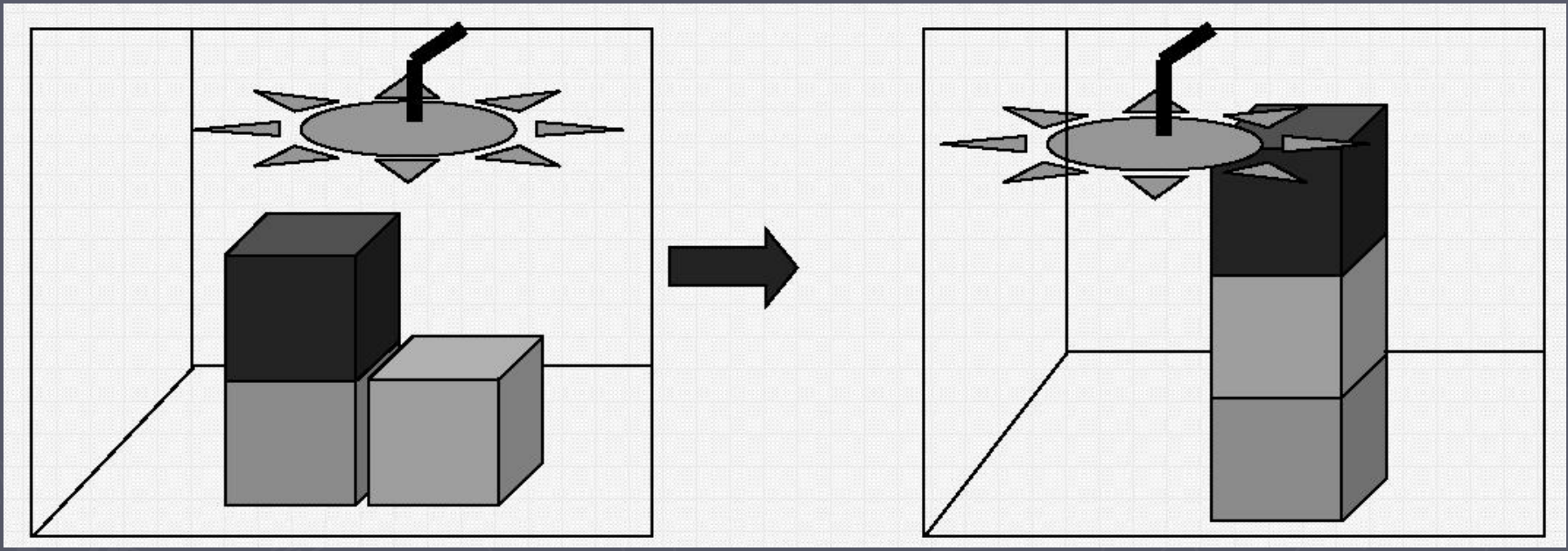
- A célok elérése feltételezi:
  - a világ bizonyos fokú ismeretét
  - az előrelátó, jövőre irányuló következtetés képességét

- A cselekvéstervezés problémáinak megközelítése:
  - **domain-independent-planning**
  - **domain-dependent-planning**

A robotika részfeladatai

- Robotszerkezet építése
- Cél-meghatározás
- Érzékelés, alakfelismerés (látás, hallás)
- **Tervgenerálás** (elemi műveletsorozat előállítása)
- Végrehajtás és javítás

Kockavilág



- Asztalról egy kockát felemel: pickup(x)
- Asztalra lerak egy kockát: putdown(x)
- Egy kockát egy kockára rátesz: stack(x,y)
- Egy kockát egy kockáról levesz: unstack(x,y)

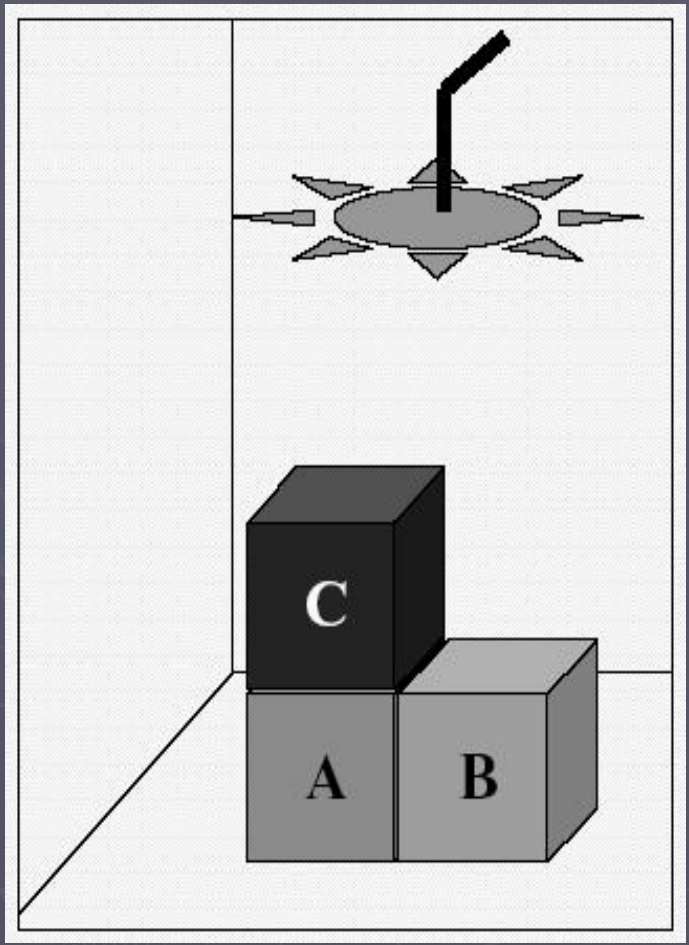
Állapot-leírás logikai állításokkal

- Elemi állítások:
  - on(x,y) egy kocka egy másik kockán van
  - clear(x) egy kocka teteje üres
  - ontable(x) egy kocka az asztalon van
  - handempty robotkar szabad
  - holding(x) robotkar egy kockát tart

Szimbólumok

- Predikátum szimbólumok:
  - ontable(x), on(x,y), clear(x), holding(x), handempty
- Függvény szimbólumok:
  - pickup(x), putdown(x), stack(x,y), unstack(x,y)

Példa



Pozitív literálok konjunkciója:

handempty /\

clear(C) /\

on(C,A) /\ clear(B) /\

ontable(A) /\ ontable(B)

Műveletek leírása

- Minden F(x) művelethez megadunk egy:
  - Előfeltétel lista — röviden P
  - Törlési lista — röviden D
  - Bővítési lista — röviden A

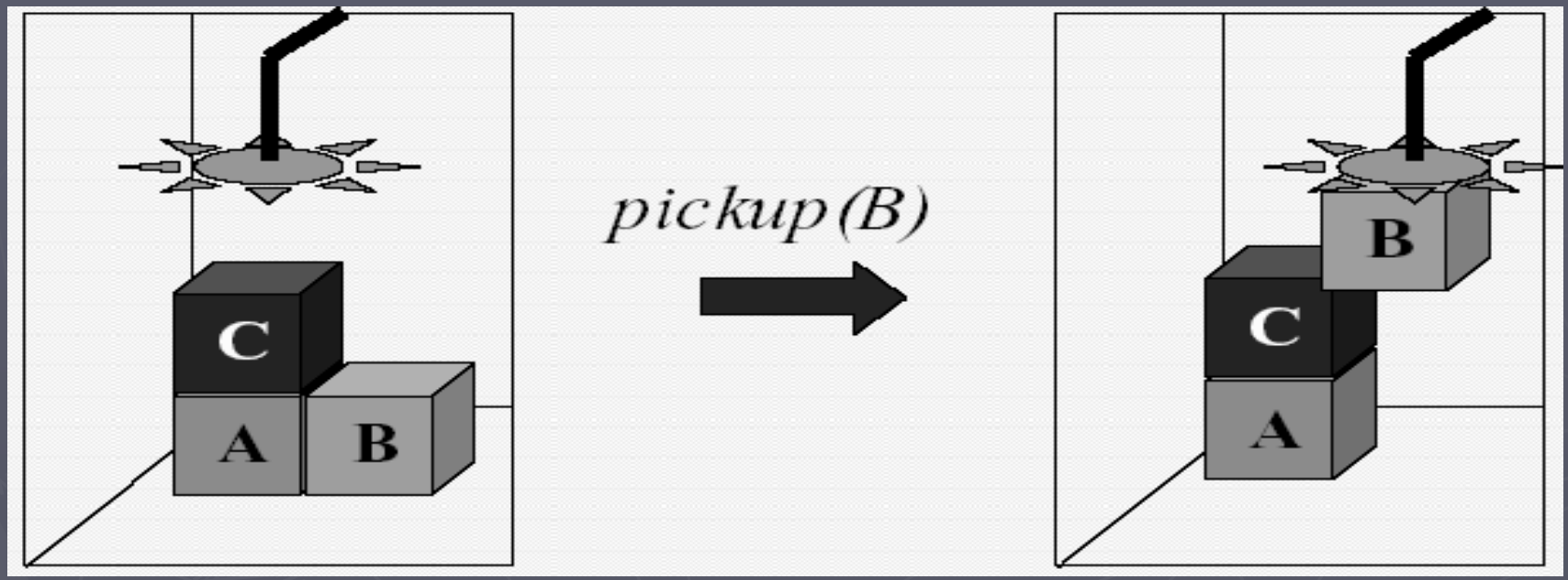
Pl: pickup(x):

P: ontable(x), clear(x), handempty

D: ontable(x), clear(x), handempty

A: holding(x)

Példa



handempty,
clear(C),
on(C,A), clear(B),
ontable(A), ontable(B)

holding(B),
clear(C),
on(C,A),
ontable(A)

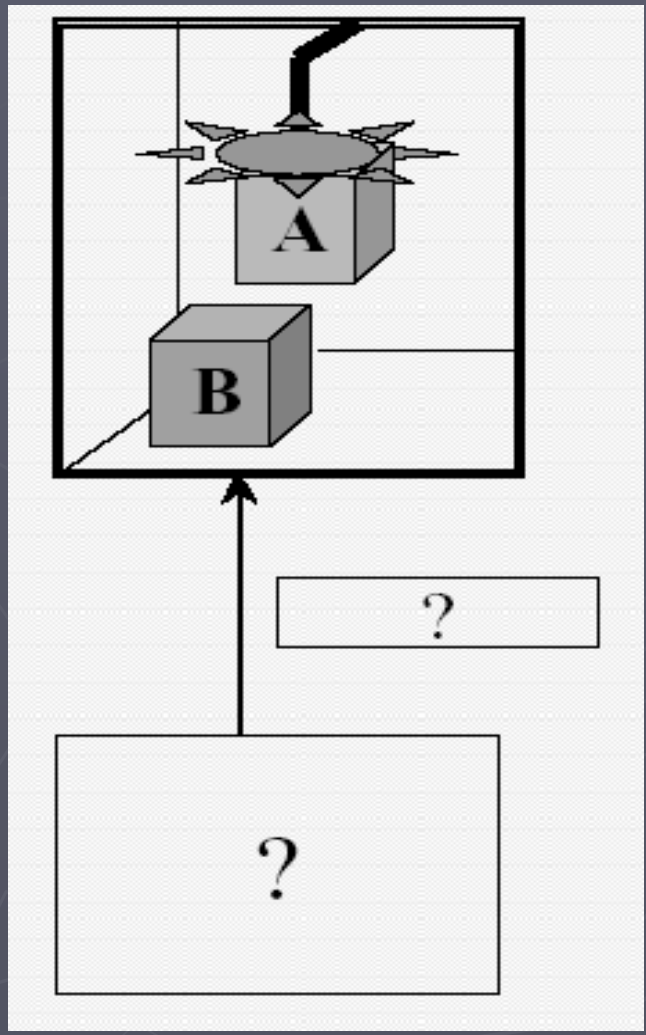
Példa további műveletekre

- putdown(x)
  - P: holding(x)
  - D: holding(x)
  - A: ontable(x), clear(x), handempty
- stack(x,y)
  - P: holding(x), clear(y)
  - D: holding(x), clear(y)
  - A: on(x,y), clear(x), handempty
- unstack(x,y)
  - P: on(x,y), clear(x), handempty
  - D: on(x,y), clear(x), handempty
  - A: holding(x), clear(y)

Megoldás állapottér-reprezentációval

- állapotgráf
- megoldási út előállítása előre vagy visszafelé haladó kereséssel
  - visszalépéses
  - gráfkeresés
- heurisztika

Redukció a tervgenerálásban



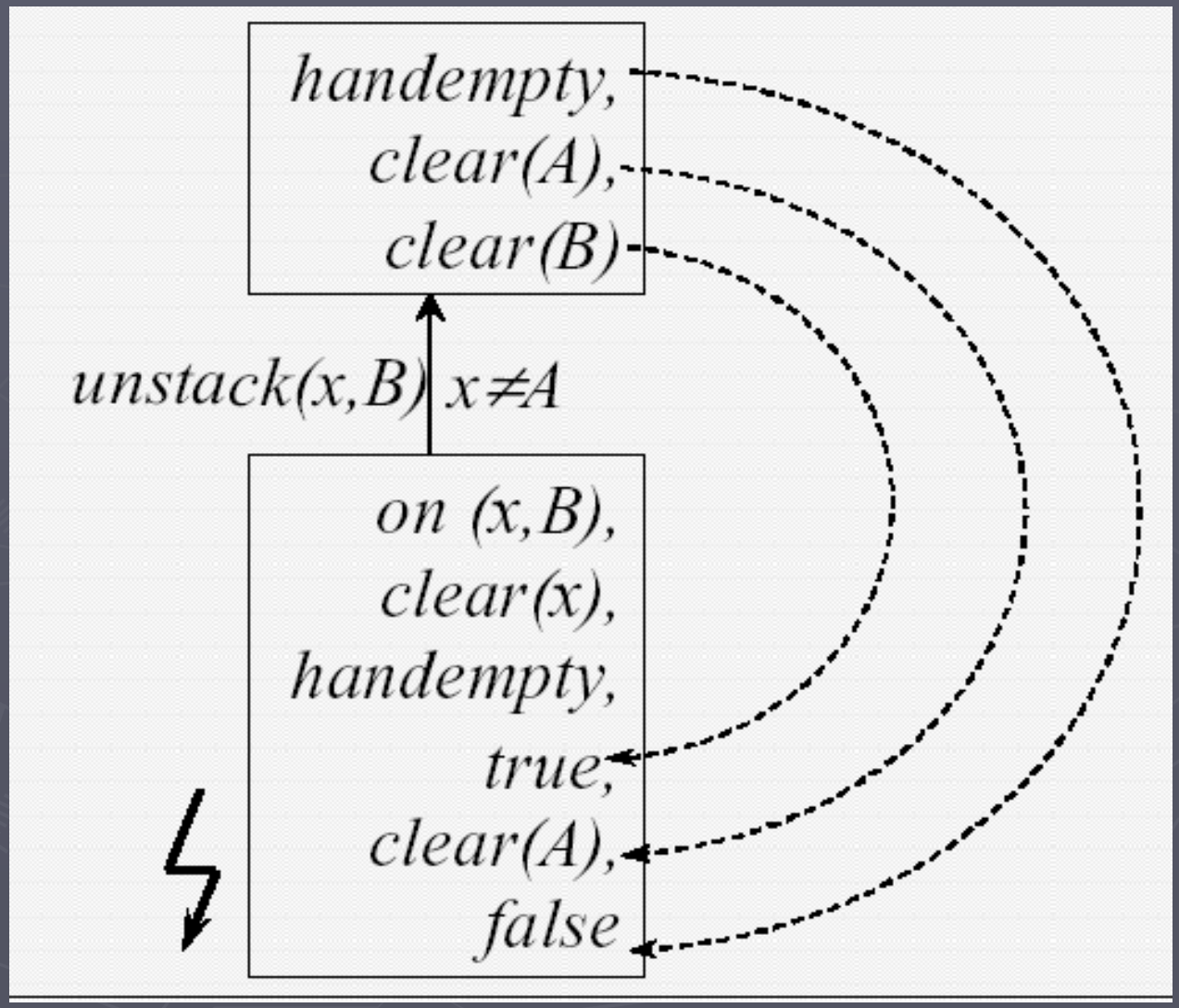
Állapot redukálása

- Kiválasztunk egy megvalósítandó *L* literált.
- Keresünk egy olyan $M = F(x)\theta$ műveletet, amely bővítési listáján ez a literál szerepel:

$$L \in A$$

- Kiszámoljuk a megelőző állapotot:
  - Vesszük a művelet előfeltételeit : *P*
  - Megvizsgáljuk, hogy a célállapot literáljainak milyen formában kell jelen lenni a megelőző állapotban

Példa


handempty,
clear(A),
clear(B)
unstack(x,B)
x≠A
on (x,B),
clear(x),
handempty,
true,
clear(A),
false

Regresszió

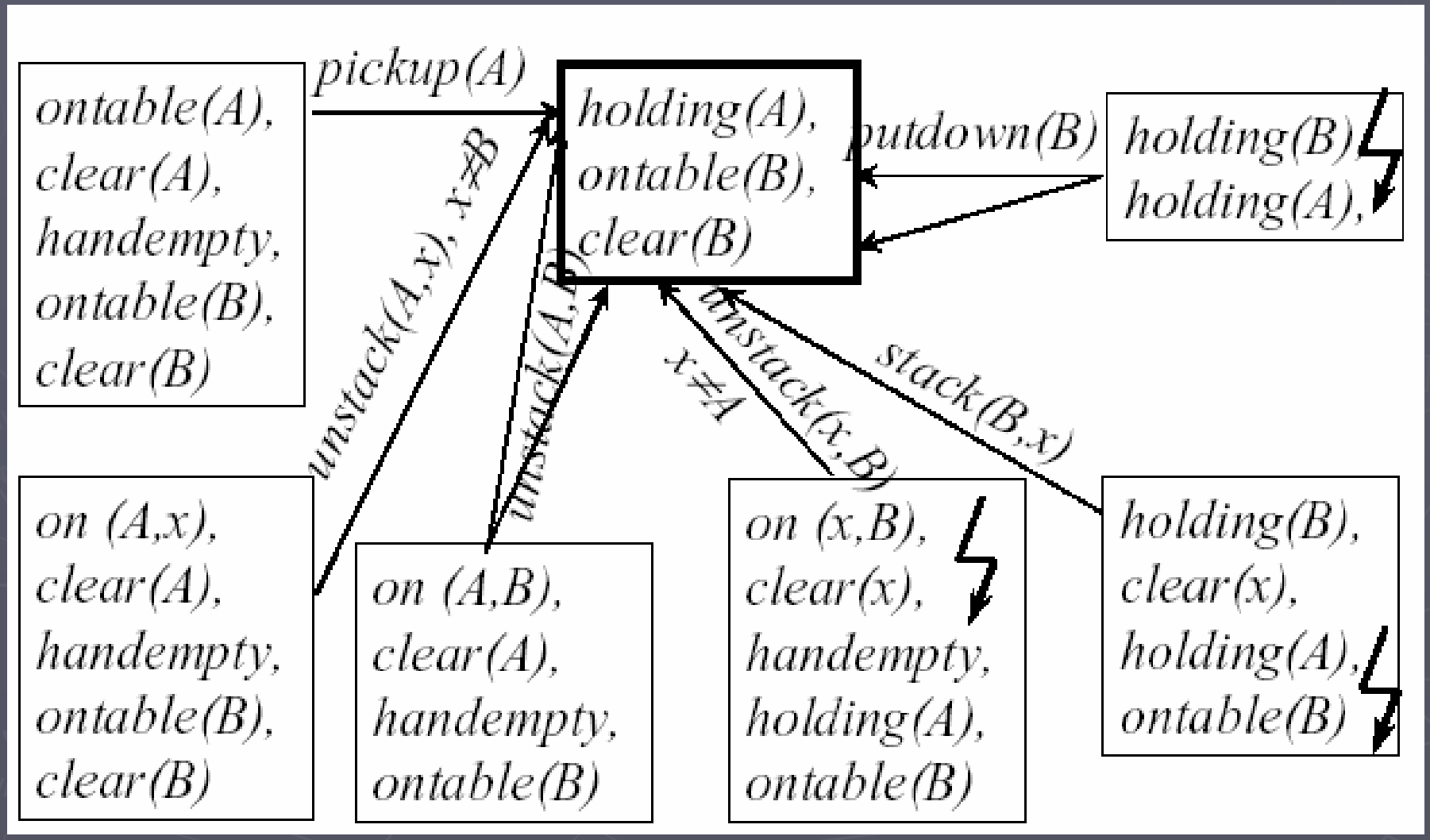
- Az L literálnak az M műveletre vett regressziója (elődje):

$$R[L;M] = \begin{cases} L & \text{ha } L \notin A \text{ és } L \notin D \\ true & \text{ha } L \in A \\ false & \text{ha } L \in D \end{cases}$$

- Az M műveletre tartózó redukciós művelet:

$$B(a) = P \wedge R[L;M] \quad (\wedge L \in a)$$

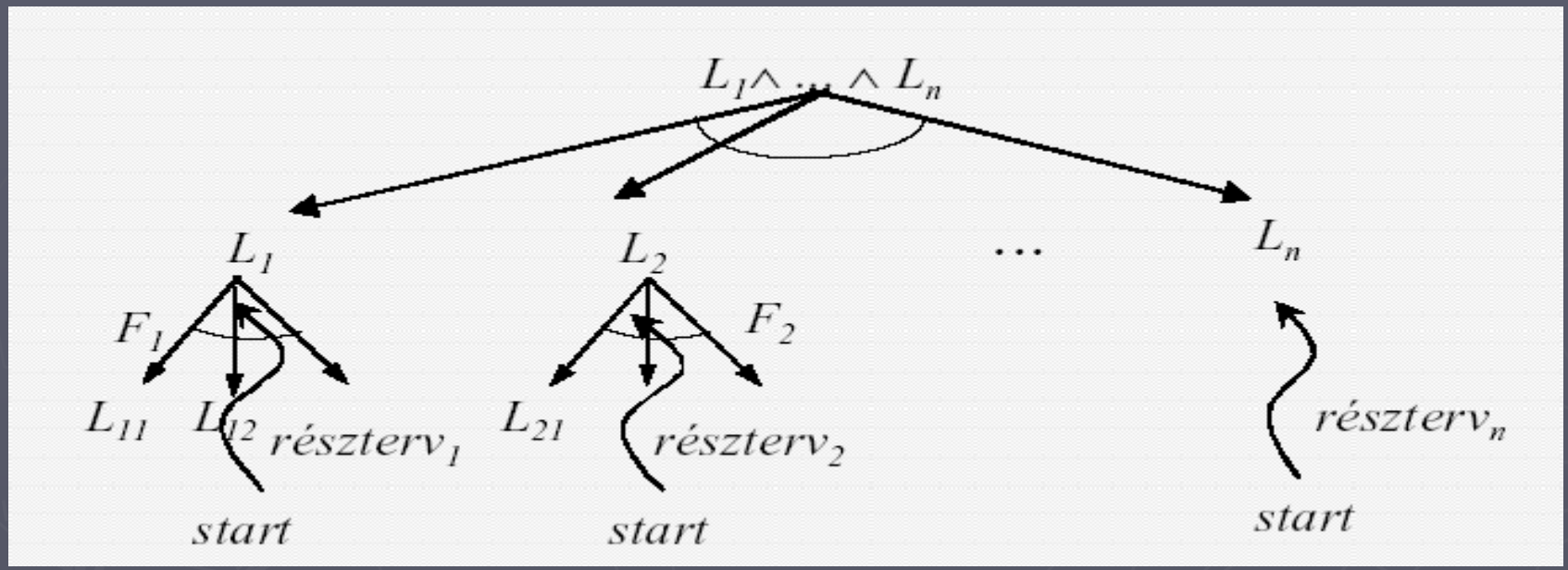
Példa


ontable(A),
clear(A),
handempty,
ontable(B),
clear(B)
pickup(A)
holding(A),
ontable(B),
clear(B)
putdown(B)
holding(B)
holding(A),
unstack(A,x), x≠B
unstack(A,B)
x≠A
unstack(x,B)
stack(B,x)
on (A,x),
clear(A),
handempty,
ontable(B),
clear(B)
on (A,B),
clear(A),
handempty,
ontable(B)
on (x,B),
clear(x),
handempty,
holding(A),
ontable(B)
holding(B),
clear(x),
holding(A),
ontable(B)

Tervgenerálás dekompozícióval

- Egy összetett L^1 /\ … /\ L^n célt tényezőnként (célliterálonként) valósítunk meg.
- Egy L literál megvalósítása olyan művelettel történik, amely bővítési listáján (megfelelő változó behelyettesítés mellett) szerepel az L. Így az L literál megvalósítását visszavezetjük a művelet előfeltételének (részcél) megvalósítására.
- A részcélokat újra és újra dekomponáljuk, amíg olyan literálokhoz nem jutunk, amelyek teljesülnek a *start*-ban.

Tiszta dekompozició



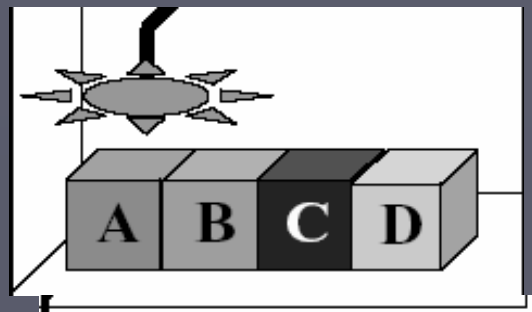
-A gráf közönséges ágait hívjuk résztervnek.

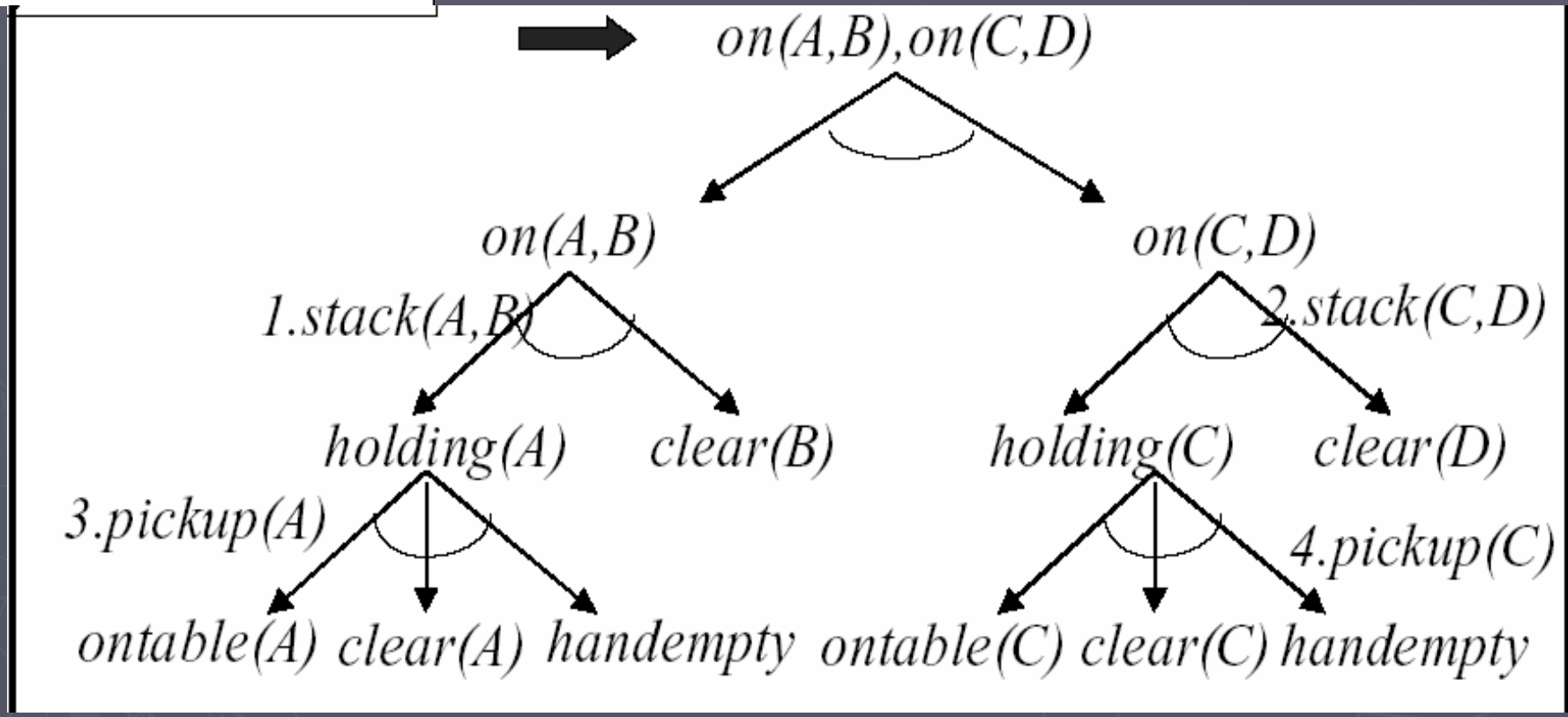
-Megoldás : *részterv1*, ..., *résztervn* valamilyen sorrendje

Dekompoziciós reprezentáció

- Probléma leírás : *start –> állapot*
- Kiinduló probléma : *start –> cél*
- Egyszerű probléma : *start –> L*, ha *L ∈ start*
- Operátorok :
  - A *start –> L^1 /\ … /\ L^n* visszavezethető a *start –> L^1, …, start –> L^n* problémákra.
  - A *start –> L* visszavezethető a *start –> P* problémákra, ahol *M* olyan művelet, hogy az *L ∈ A*

Példa



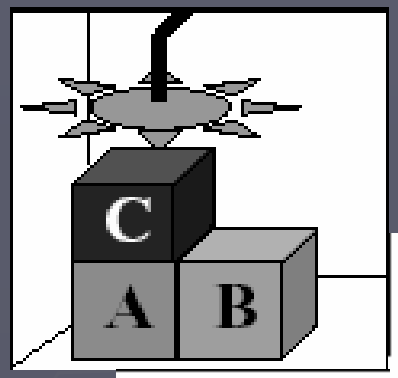


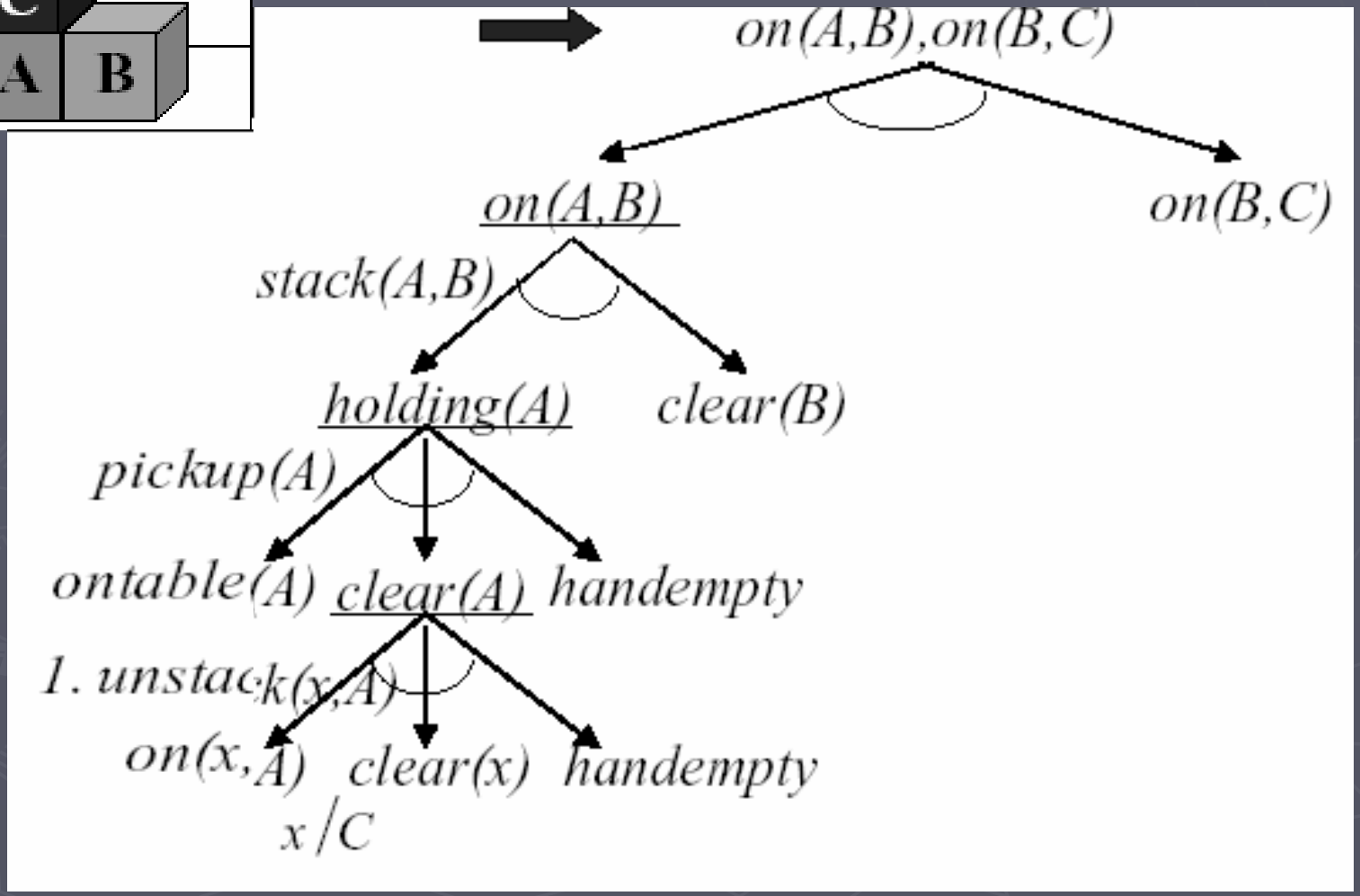
Megoldás : *3, 1, 4, 2* vagy *4, 2, 3, 1*

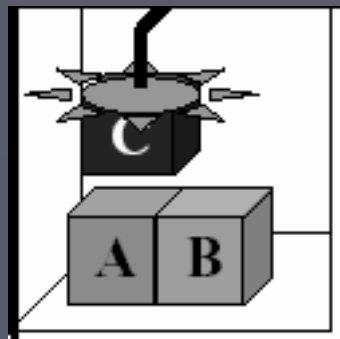
Döntési pontok, heurisztikák

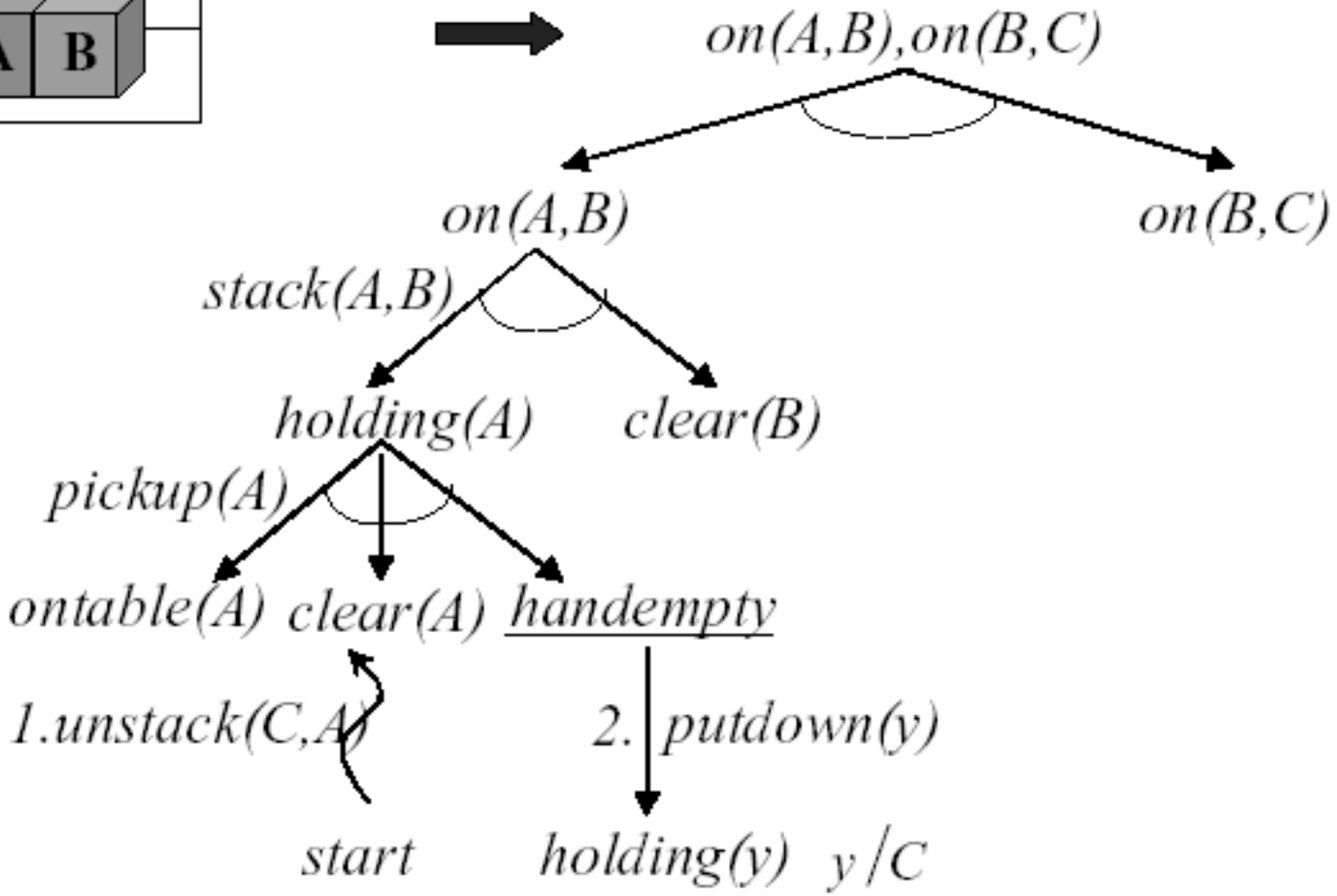
- Melyik cél-literált valósítsuk meg először?
- Melyik műveletet válasszuk?
- Melyik változó-helyettesítést alkalmazzuk?

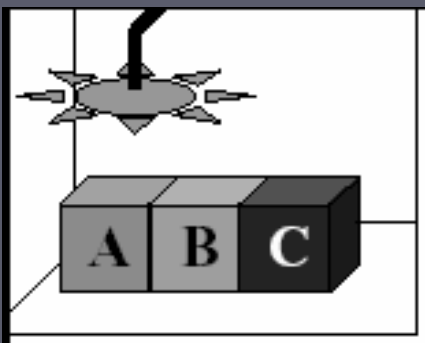
Példa

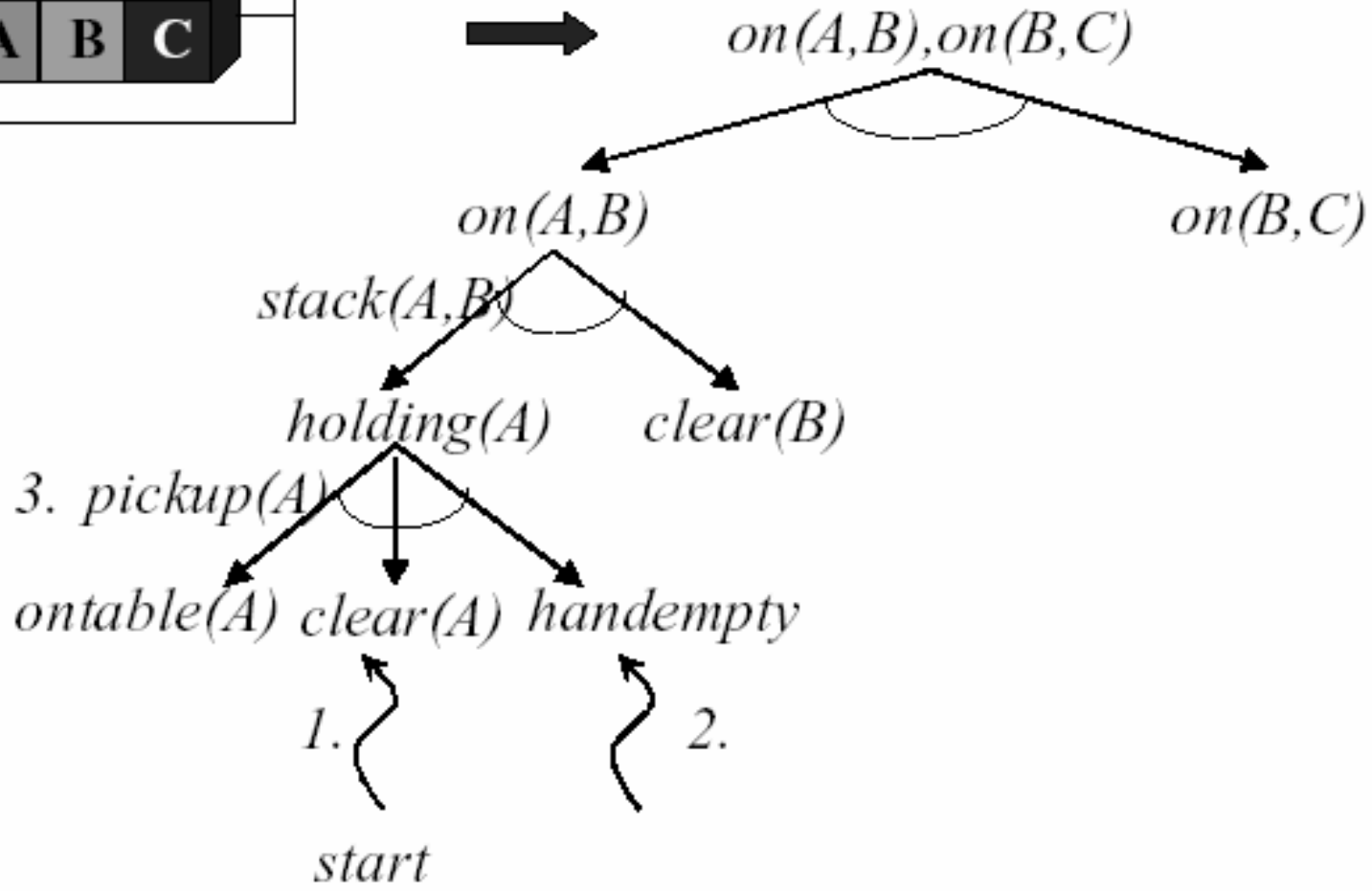

C
A
B

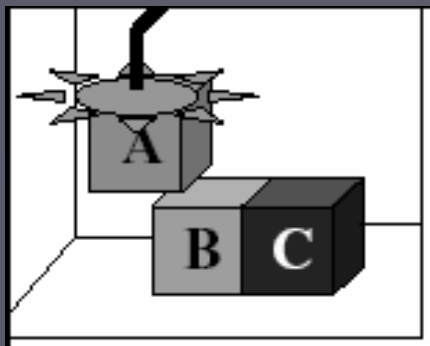

on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
pickup(A)
ontable(A)
clear(A)
handempty
1. unstack(x,A)
on(x,A)
clear(x)
handempty
x/C


C
A
B


on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
pickup(A)
ontable(A)
clear(A)
handempty
1.unstack(C,A)
2. putdown(y)
start
holding(y)
y/C


A
B
C


on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
3. pickup(A)
ontable(A)
clear(A)
handempty
1.
2.
start



on(A,B),on(B,C)

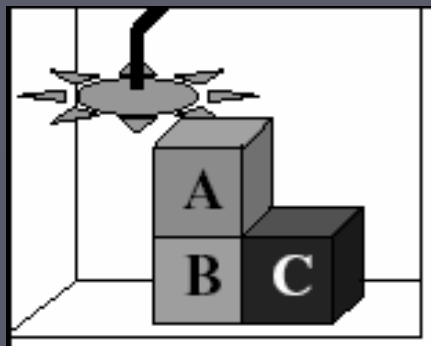
on(A,B) *on(B,C)*

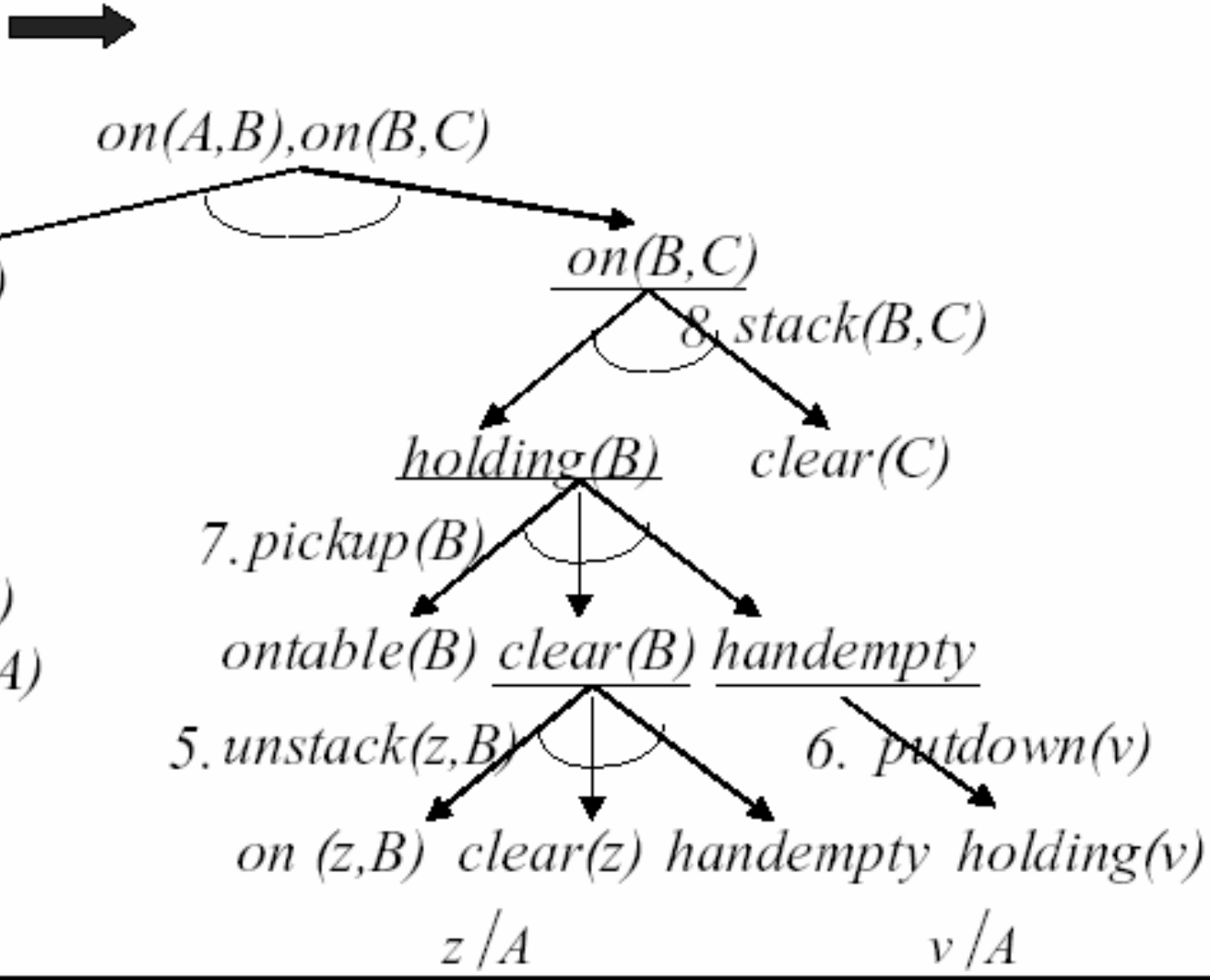
4. stack(A,B)

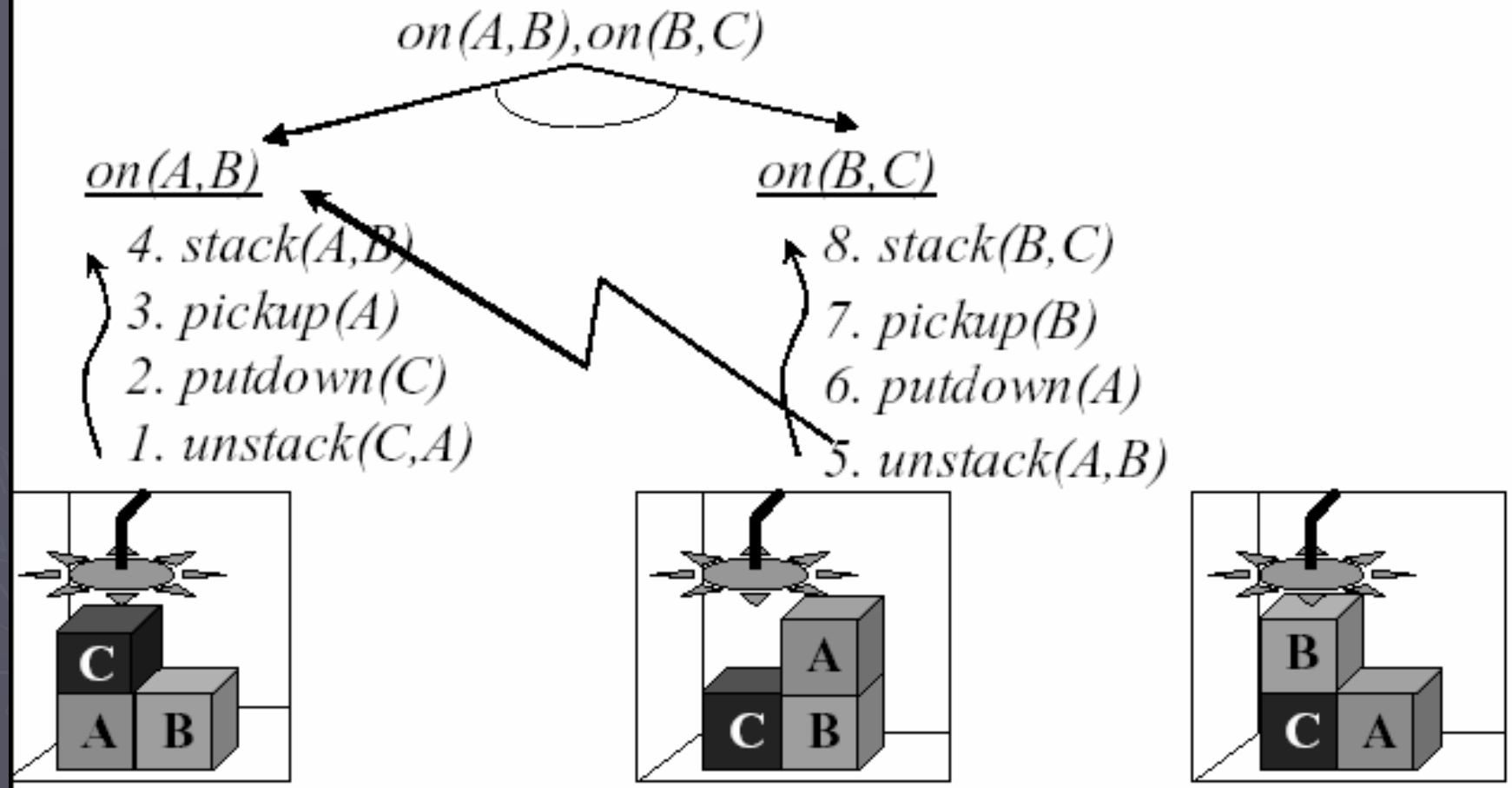
holding(A) *clear(B)*

3. pickup(A)
2. putdown(C)
1. unstack(C,A)

start


A
B
C


on(A,B),on(B,C)
on(A,B)
on(B,C)
8. stack(B,C)
4. stack(A,B)
3. pickup(A)
2. putdown(C)
1. unstack(C,A)
holding(B)
clear(C)
7. pickup(B)
ontable(B)
clear(B)
handempty
5. unstack(z,B)
6. putdown(v)
on (z,B)
clear(z)
handempty
holding(v)
start
z /A
v /A


on(A,B),on(B,C)
on(A,B)
4. stack(A,B)
3. pickup(A)
2. putdown(C)
1. unstack(C,A)
on(B,C)
8. stack(B,C)
7. pickup(B)
6. putdown(A)
5. unstack(A,B)
C
A
B
A
C
B
B
C
A

I.STRIPS

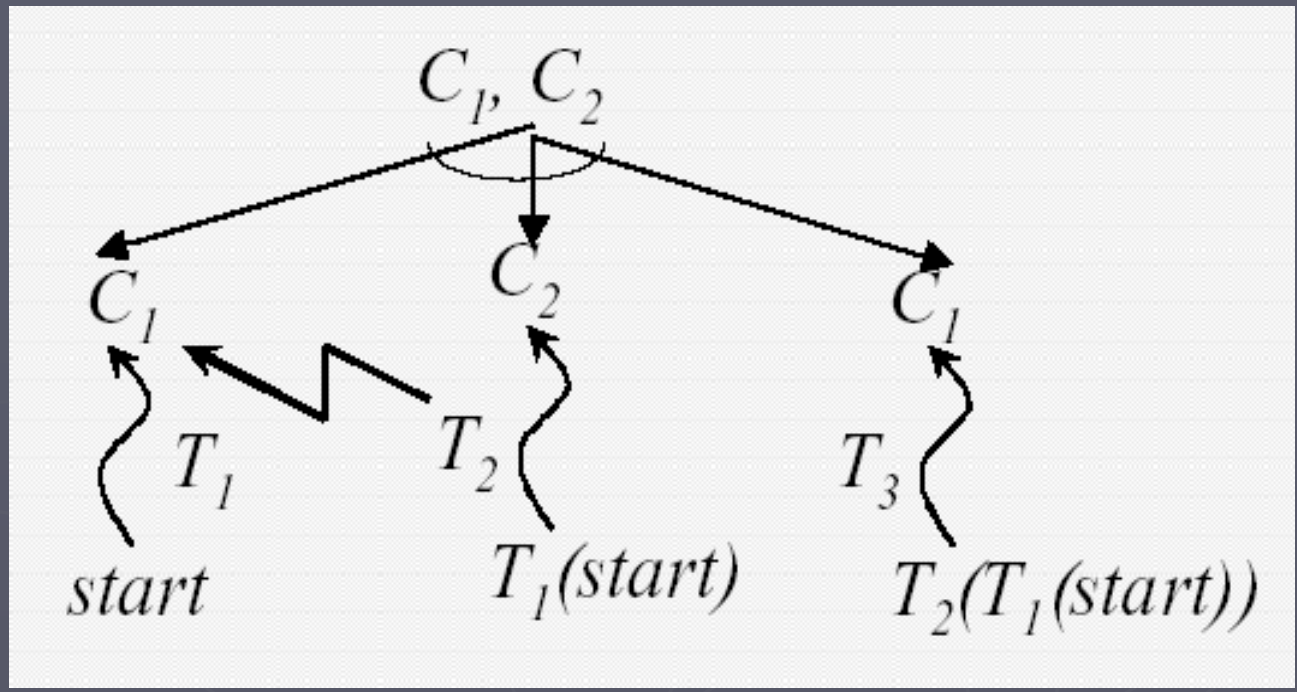
- Egyedül az változik, amit egy akció hatásaként megadunk – ez először a STRIPS (Stanford Research Institute Problem Solver) tervezőrendszerben jelent meg.
- PDDL (Planning Domain Definition Language) reprezentációs nyelvet használja.

Főbb elemei

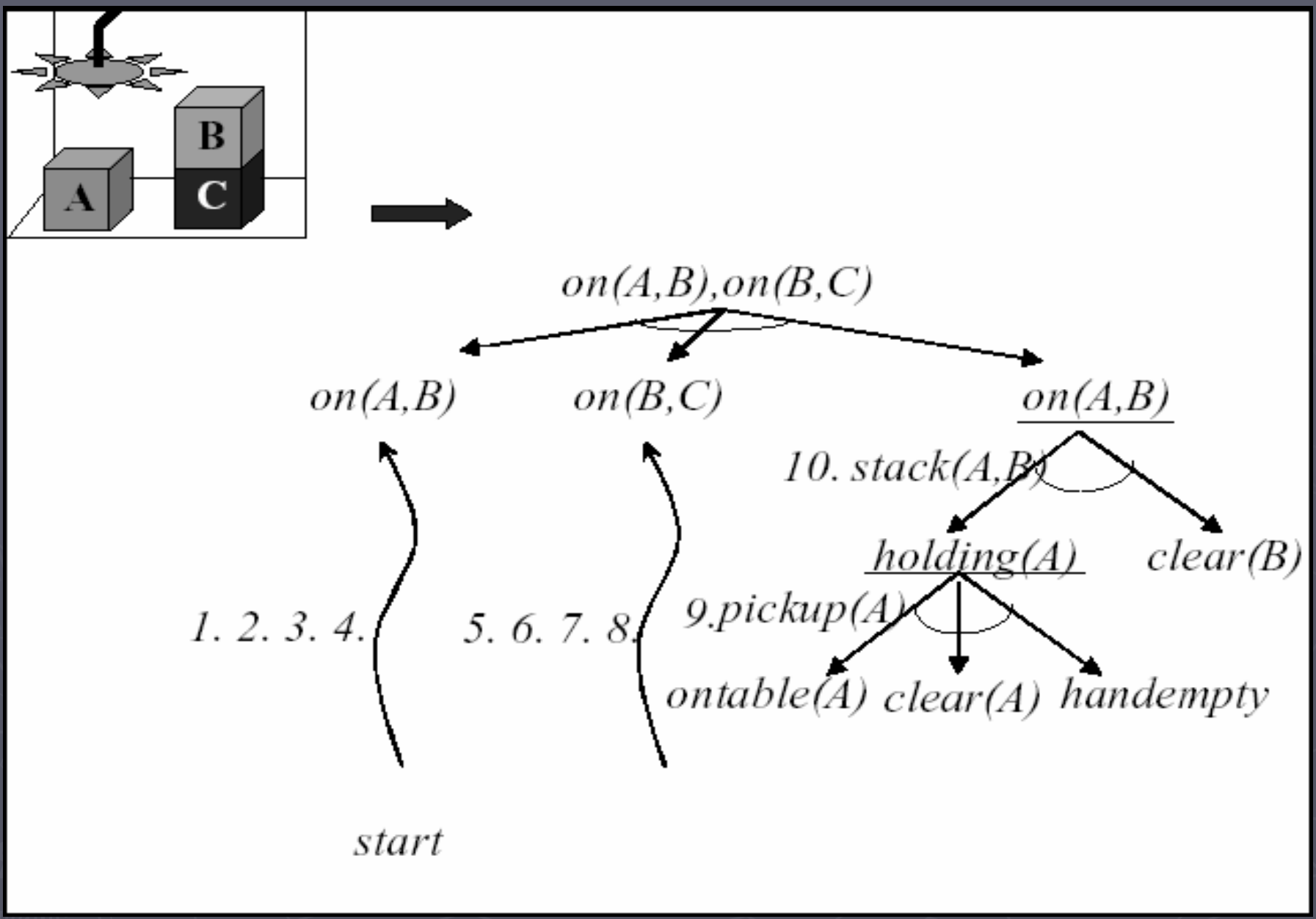
- A világ bármely állapota a benne igaz elemi logikai állítások **konjunkciójaként** írható le; ami nem szerepel a leírásban a zárt világ számára hamis.
- A célt azon elemi állítások konjunkciója adja meg, amelyekről elvárjuk, hogy bármely céllálapotban igazak legyenek.

- Az operátorokat előfeltételeik és hatásaik határozzák meg
  - Az operátoroknak vannak változói
  - Előfeltételek konjunktív logikai kifejezések – hatásaik megadják, hogy milyen állítások válnak igazzá, ill. hamissá az operátor végrehajtását követő állapotban
  - Az akciópéldány végrehajtása előtt a változókat le kell kötni a feltétel oldalon

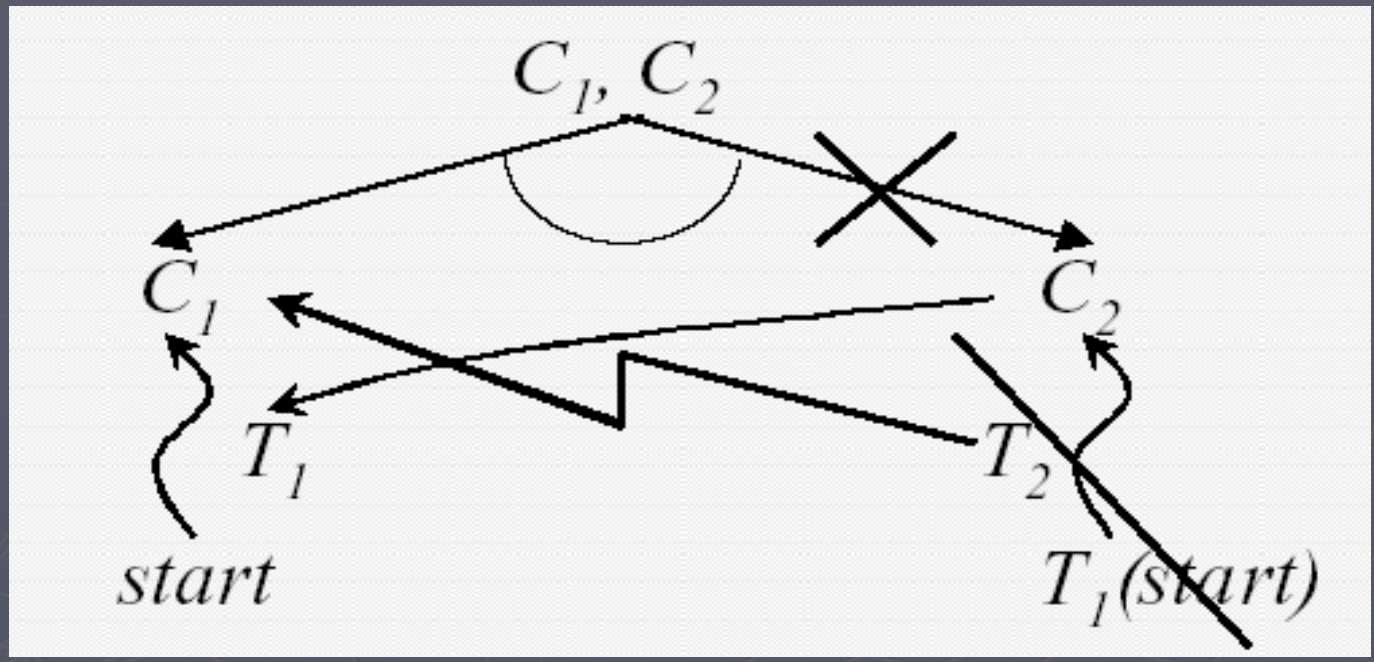
- Mi határozza meg?
  - Egy kezdeti állapot
  - Cél állapotok specifikációja
  - Action-ok halmaza, amelyek mindegyikét meghatározzák az előfeltételek és az utófeltételek
- Lényege :
  - operátorokba bújtatva írja le a világ összes törvényszerűségét
  - egy operátor alkalmazásának következményei: a világ törvényeit töröljük és a világ törvényeihez hozzáadunk.



- Hagyjuk, hogy egy későbbi (T2) részterv elrontson egy már megvalósított (C1) részcélt, amit később újra megvalósítunk az aktuális állapotból kiindulva.

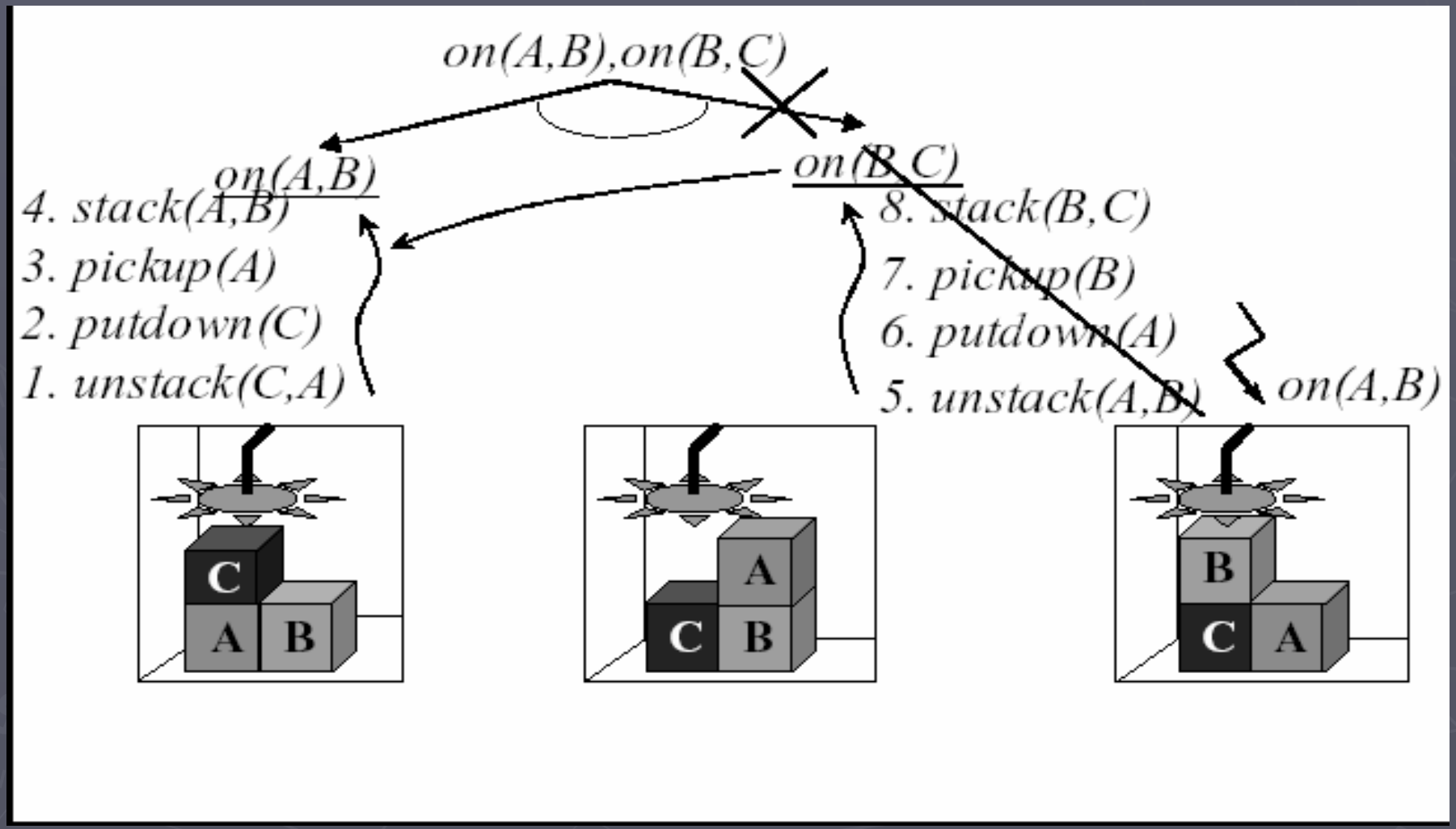

A
B
C
on(A,B),on(B,C)
on(A,B)
on(B,C)
on(A,B)
10. stack(A,B)
holding(A)
clear(B)
9.pickup(A)
ontable(A)
clear(A)
handempty
1. 2. 3. 4.
5. 6. 7. 8.
start

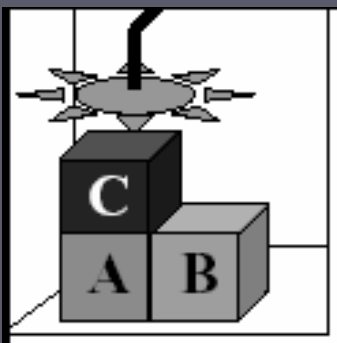
II.RSTRIPS



- Töröljük a védett C1 részcélt elrontó tervet.
- Valósítsuk C2-t a T1 részterv utolsó művelete előtt
  - Legyen ennek a műveletnek a C2 egy újabb előfeltétele
  - Feltéve, hogy a C2-t nem törli ez a művelet.

Példa


on(A,B),on(B,C)
on(A,B)
4. stack(A,B)
3. pickup(A)
2. putdown(C)
1. unstack(C,A)
on(B,C)
8. stack(B,C)
7. pickup(B)
6. putdown(A)
5. unstack(A,B)
on(A,B)
C
A
B
A
C
B
B
C
A



on(A,B),on(B,C)

on(A,B)

on(B,C)

stack(A,B)

holding(A)

clear(B)

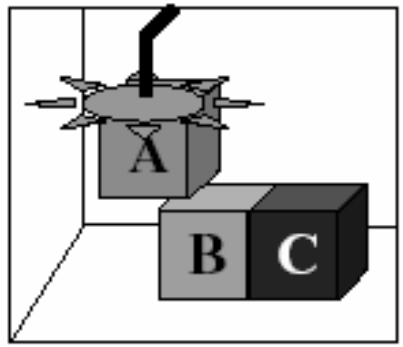
on(B,C)

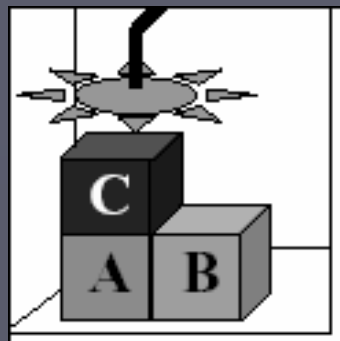
3. pickup(A)
2. putdown(C)
1. unstack(C,A)

start

stack(B,C)
pickup(B)
putdown(A)

holding(A)





on(A,B),on(B,C)

on(A,B)

6. stack(A,B)

holding(A)

clear(B)

5. pickup(A)

ontable(A) clear(A) handempty

on(B,C)

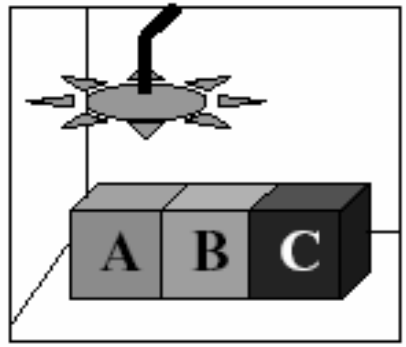
1. unstack(C,A) 2. putdown(C)

4. stack(B,C)

3. pickup(B)

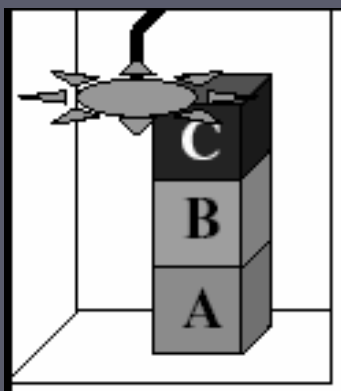
handempty ✓

start holding(C)



III.DCOMP

- 1. fázis:
  - Résztervek előállítása eredeti dekompozicióval.
- 2. fázis:
  - A felesleges műveletek törlése, sorrendi megkötések fellállítása a műveletekre, majd a résztervek összefésülése.
- 3. fázis:
  - Ha az összefésülés eredménye egy ütközéses terv, akkor új műveletek beillesztésével az ütközéseket megszüntetjük.



Példa

on(A,B),on(B,C)

on(A,B) A=1

on(B,C) A=2

A=2,4,7,8 D=5,6

1.stack(A,B)

clear(B)

2.stack(B,C)

holding(A)

4.unstack(C,B)

clear(C) D=4,7,8

3.pickup(A) A=3

on(C,B) clear(C) handempty D=2,7,8 A=2

ontable(A) clear(A) handempty

holding(B) A=5 D=3,5,6,7,8

6.unstack(B,A) A=5,6 A=2 D=4,5,6,7,8

5.unstack(B,A) A=4,7,8 D=1,6

on(B,A) clear(B) handempty

8.unstack(C,B)

7.unstack(C,B) A=1 D=3,4,6,7,8

on(C,B) clear(C) handempty

on(B,A) clear(B) handempty

on(C,B) clear(C) handempty

Sorrendi megkötések

- Természetes sorrend (1. fázis résztervei)
- Mely műveletek nem követhetik egymást feltétel nélkül?
  - Az *i* művelet után nem hajtható végre a *j*, ha az *i* törli a *j* előfeltétel-literáljainak egyikét.
  - Ha *i* művelet után nem hajtható végre a *j*, akkor feltüntetjük, *j* mely előfeltételei sérülnek.
  - Keresünk olyan *k* műveletet, amely az *i* után és a *j* előtt végrehajtva előállítja a sérült feltételt.

Példa

- A 4. művelet (unstack(C,B)) végrehajtása után:
  - holding(C), clear(B), on(B,A), ontable(A)
- A részcél ami nekünk kell:
  - on(B,A), clear(B), handempty, ontable(A)
- Megoldás:
  - putdown(C)

IV.NONLIN

- Párhuzamosítjuk a DCOMP első két fázisát:
  - Nem akarjuk a műveleteket egyetlen szálra felfűzni
  - Amint egy művelet megjelenik, azonnal megmondjuk, hogy mely, már meglevő műveletekkel ütközhet, és ilyenkor sorrendi kötéseket teszünk, illetve jelöljük azokat az előfeltétel literálokat, amelyeket a vizsgált két művelet végrehajtása között kell megvalósítani.

Logikai reprezentációk

- Tény
  - Kezdőállapot leírása
- Szabályok
  - Műveletekből származó *P -> A* állítások
- Cél
  - Célállapot leírása
- Bizonyítandó:
  - Kezdőállapot, műveleti szabályok => célállapot

Példa

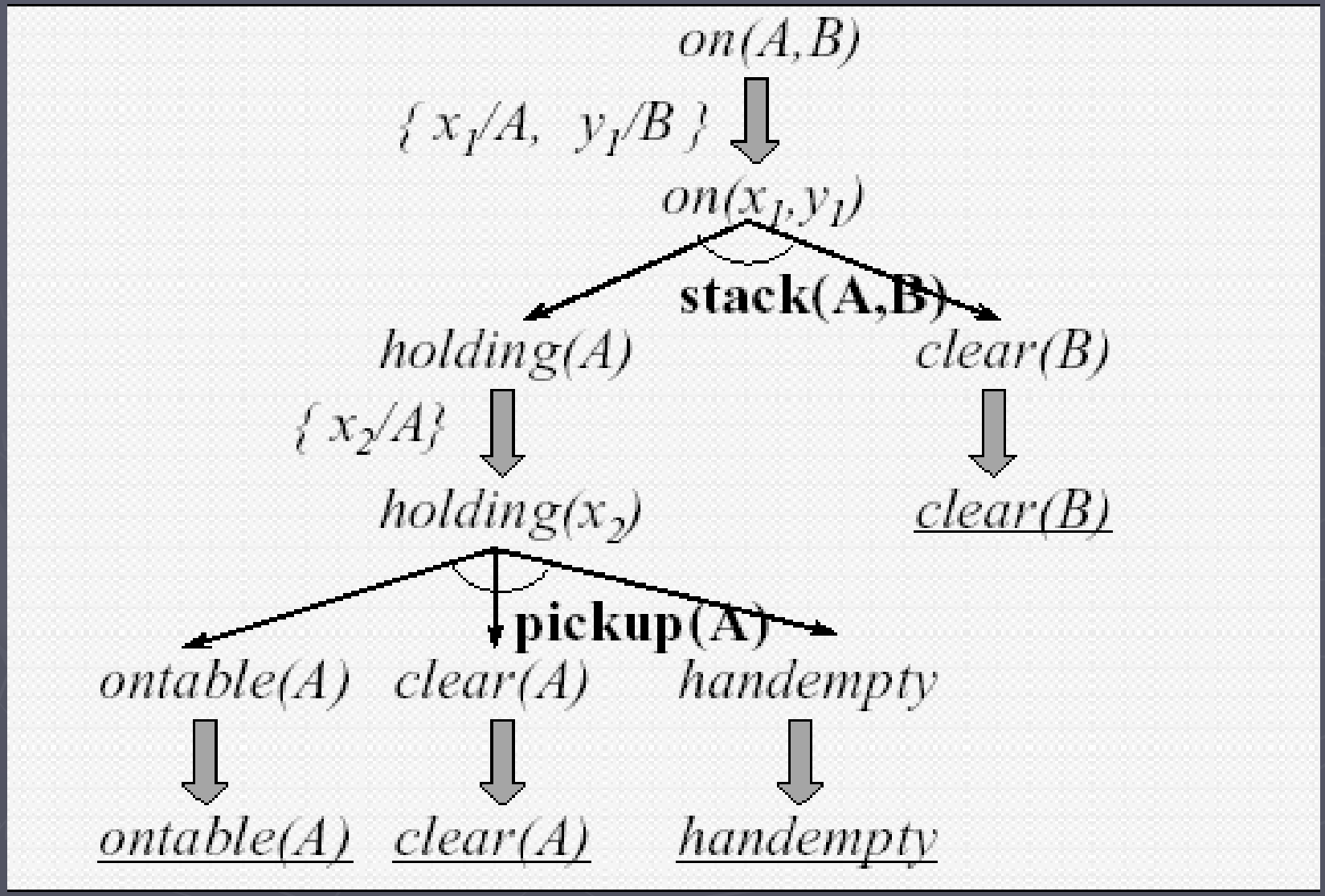
- Kezdőállapot:
  - handempty, clear(A), clear(B), ontable(A), ontable(B)
- Célállapot:
  - on(A,B)

- Szabályok:
  - ∀x ∀y (holding(x) /\ clear(y) ->
    on(x,y) /\ clear(x) /\ handempty)
  - ∀x ∀y (on(x,y) /\ (clear(x) /\ handempty ->
    holding(x) /\ clear(y))
  - ∀x (holding(x) ->
    ontable(x) /\ clear(x) /\ handempty)
  - ∀x (ontable(x) /\ clear(x) /\ handempty ->
    holding(x))

Visszafelé haladó szabályalapú reprezentáció

- Tény:
  - handempty, clear(A), clear(B), ontable(A), ontable(B)
- Szabályok:
  - ∀x (ontable(x) /\ clear(x) /\ handempty -> holding(x))
  - ∀x ∀y (holding(x) /\ clear(y) -> on(x,y))
  - ∀x ∀y (holding(x) /\ clear(y) -> clear(x))
  - ∀x ∀y (holding(x) /\ clear(y) -> handempty)
  - ...
- Cél:
  - on(A,B)

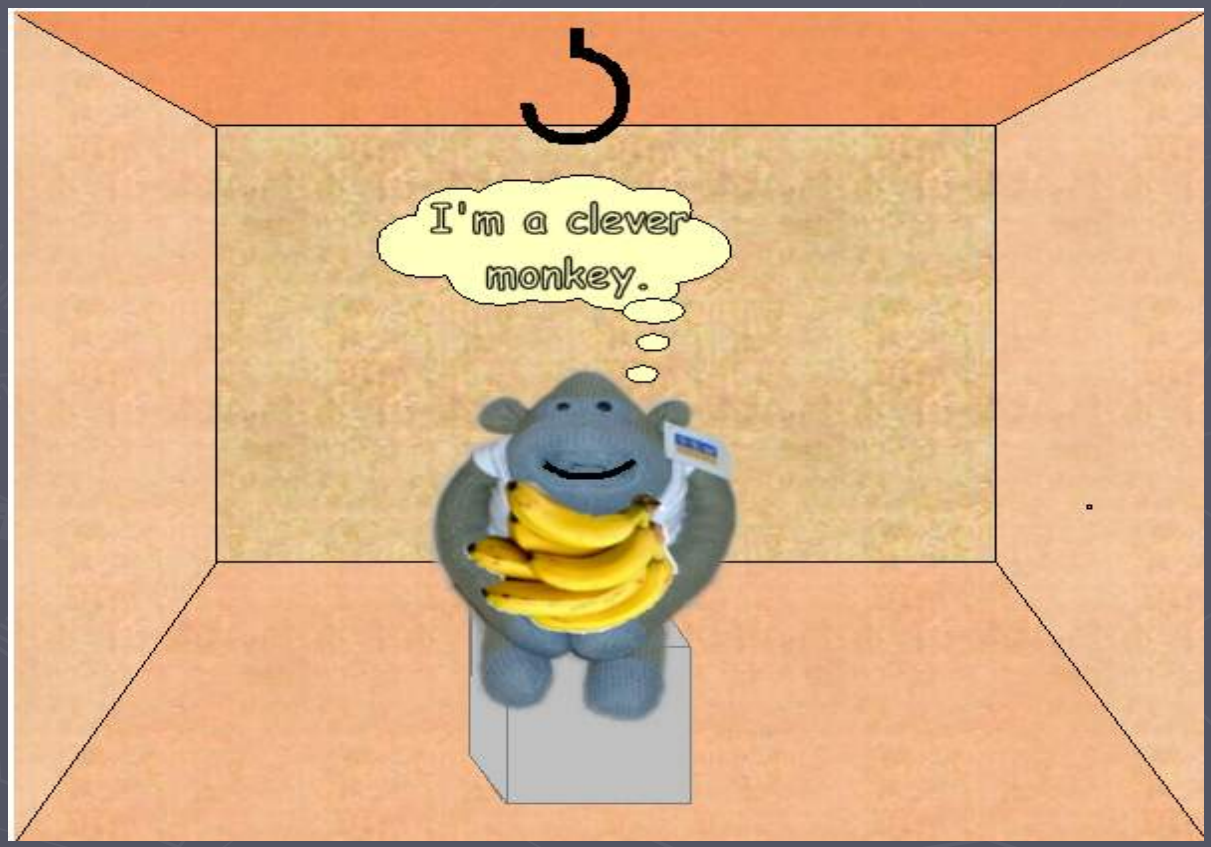
Szabályalapú következtetés



Majom és banán példa (PROLOG)

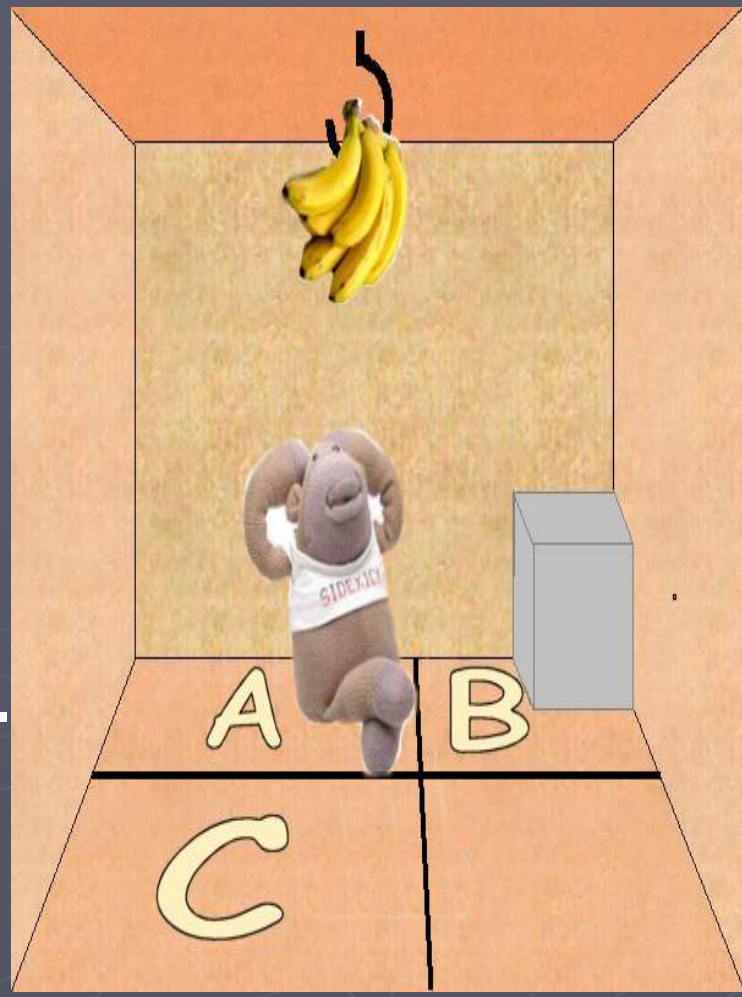
- Egy szobában van egy majom, a plafonon csüng 10 banán, a szoba sarkában van egy doboz. A majom éhes és kell neki a banán. Mi a teendő? Kell egy terv!

- Néhány sikertelen probálkozás után a majom elmegy a dobozhoz, odanyomja a banán alá, felmászik rá, leszedi a banánt és megeszi.

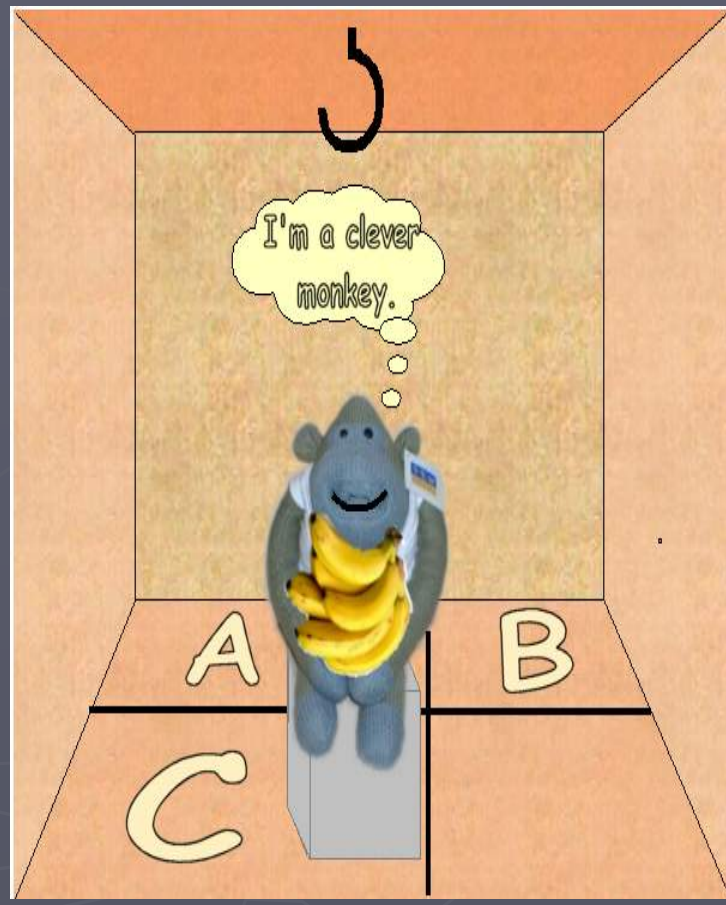


Kezdeti- és cél-állapot

- Kezdeti állapot:
  - `on(monkey, floor),`
  - `on(box, floor),`
  - `at(monkey, a),`
  - `at(box, b),`
  - `at(bananas, c),`
  - `status(bananas, hanging).`



- Cél állapot:
  - `on(monkey, box),`
  - `on(box, floor),`
  - `at(monkey, c),`
  - `at(box, c),`
  - `at(bananas, c),`
  - `status(bananas, grabbed).`



- Ezeket tudva, hogyan valósítjuk ezt meg?

Forrásanyag

- http://quasar.inf.elte.hu/
- http://people.inf.elte.hu/gt/mi/mi.html
- www.mestersegesintelligencia.hu
- http://cse.stanford.edu/class/sophomore-college/projects-98/robotics/
- http://www-users.cs.umn.edu/~gini/motion.html
- http://www.inf.ed.ac.uk/teaching/courses/aipp/