TERVGENERÁLÁS

1. Állapottér-reprezentáció
2. Probléma redukció
3. Probléma dekompozíció
4. Logikai reprezentáció

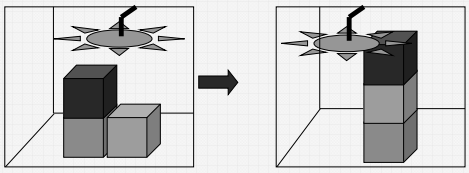
1

Robotika részfeladatai

- robot-szerkezet építése
- cél-meghatározás
- érzékelés, alakfelismerés (látás, hallás, stb.)
- tervgenerálás (elemi műveletsorozat előállítása)
- végrehajtás és korrigálás

2

Kocka világ



- 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)*

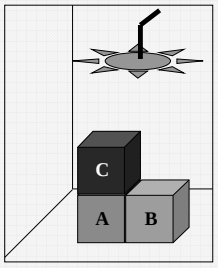
3

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

4

Példa



pozitív literálok konjunkciója:

handempty ∧

clear(C) ∧

on(C,A) ∧ *clear(B)* ∧

ontable(A) ∧ *ontable(B)*

5

Állapot-leírás tulajdonságai

- Konkrét vagy általános állapot-leírás.
  - Változók használata.
- Teljes vagy hiányos állapot-leírás.
  - Hiányos leírás lehet egyértelmű vagy többértelmű.
- Helyes vagy ellentmondásos állapot-leírás.
  - Ellenőrzés: konstrukcióval vagy logikai következtetéssel.

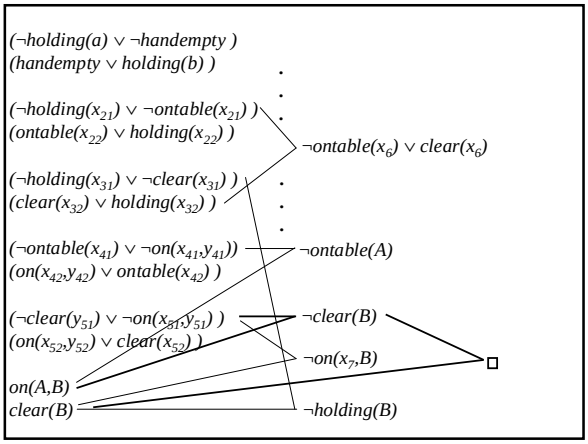
6

Ellentmondásos állapot

- Egy állapot-leírás lehet helyes vagy ellentmondásos
  - Az *on(A,B) ∧ clear(B)* állapot ellentmondásos
- Egy állapot ellentmondása eldönthető
  - konstrukcióval
    - *teljessé alakítjuk*
    - *ontable, on, holding* „végrehajtása”
    - *clear, handempty* ellenőrzése
  - rezolúcióval, megfelelő axiómarendszer alapján
    - Pl.: *on(x,y) → ¬clear(y) ∧ ¬ontable(x) ∧ ¬holding(x) ∧ ¬holding(y)*

Példa

- Helyes-e az *on(A,B) ∧ clear(B)* állapot-leírás az alábbi axiómarendszer ismeretében?
  - $\exists x$ *holding(x)* $\leftrightarrow \neg$ *handempty*
  - $\forall x$ *(holding(x)*$\leftrightarrow \neg$*ontable(x))*
  - $\forall x$ *(holding(x)*$\leftrightarrow \neg$*clear(x))*
  - $\forall x$ *(ontable(x)*$\leftrightarrow \neg$*on(x,y))*
  - $\forall x$ *(clear(y)* $\leftrightarrow \neg$*on(x,y))*
- Másként feltéve a kérdést: kielégíthetetlen-e az a logikai rendszer, amelyben a fenti axiómákhoz hozzávesszük az *on(A,B) ∧ clear(B)* állítást?

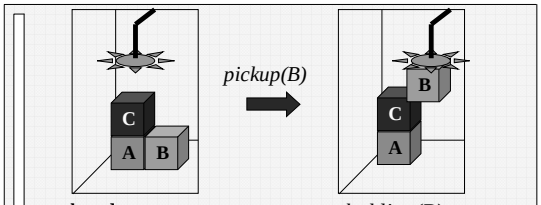


Műveletek leírása

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

Példa: *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)

Példák 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)

Megjegyzés

- A P^F nem feltétlenül azonos D^F-vel. Például, ha a robotkar egy művelete az asztalon fekvő *x* kockát felteszi az *y* kockára (*pickup(x)+stack(x,y)*):
  - *puton(x,y):*
    P^F*: ontable(x), clear(x),clear(y), handempty*
    D^F*: ontable(x), clear(y)*
    A^F*: on(x,y)*
- A D^F többnyire része P^F-nek

13

Megjegyzés

- Egy *M* művelet az *F(x)* művelet-leírásnak egy *F(x)*α alappéldánya. (α egy helyettesítés)
- Az előfeltétel igazolása, valamint a művelet eredményének kiszámolása halmazműveletek segítségével történik.
- Egy művelet teljes állapot-leírásból (konkrét állapotból) teljes állapot-leírást (konkrét állapotot) hoz létre.
- A művelet-leírás alapján hiányos és általános állapot-leírást is át lehet alakítani.

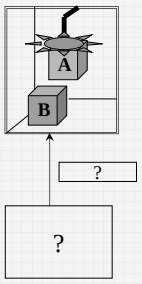
14

1. Megoldás állapottér-reprezentációval

- Állapotok, műveletek meghatároznak egy állapotgráfot, benne egy kezdő- és egy célállapot
- Megoldási út előállítása előre vagy visszafelé haladó kereséssel
  - visszalépéses
  - gráfkeresés
- Heurisztika

15

2. Redukció a tervgenerálásban



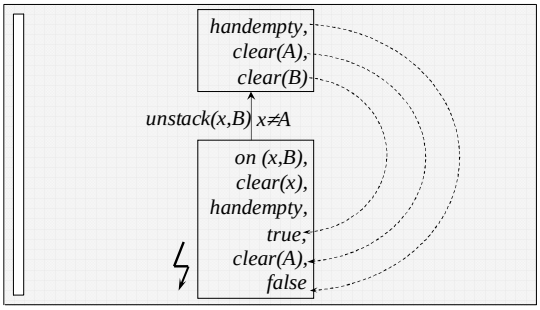
16

Állapot redukálása

- Kiválasztunk egy megvalósítandó *L* literált
- Keresünk olyan *M*=*F*(*x*)α műveletet, amely bővítési listáján ez a literál szerepel: $L \in A^M$
- Kiszámoljuk a megelőző állapotot:
  - vesszük a művelet előfeltételét: P^M
  - megvizsgáljuk, hogy a célállapot literáljainak milyen formában kell jelen lenni a megelőző állapotban

17

Példa



18

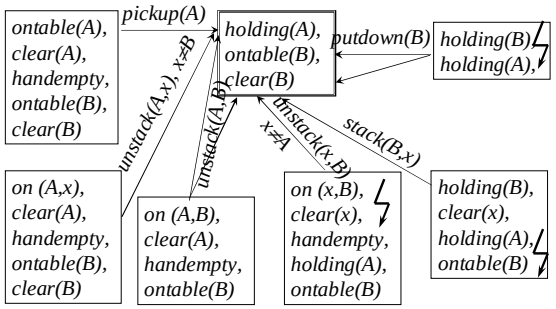
Regresszió

Az L literálnak az M műveletre vett regressziója (elődje)

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

- Az M művelethez tartozó redukciós művelet:
  - $B_M(a) = P^M \wedge \bigwedge_{L \in a} R[L; M]$

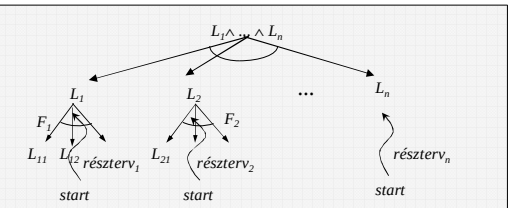
Példa



3. Tervgenerálás dekompozícióval

- Egy összetett $L_1 \wedge ... \wedge L_n$ 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 dekompozíció

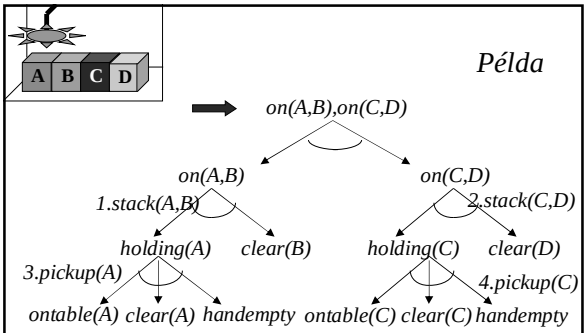


- A gráf közönséges ágait hívjuk résztervnek.
- Megoldás: *részterv*$_1$, …, *részterv*$_n$ valamilyen sorrendje

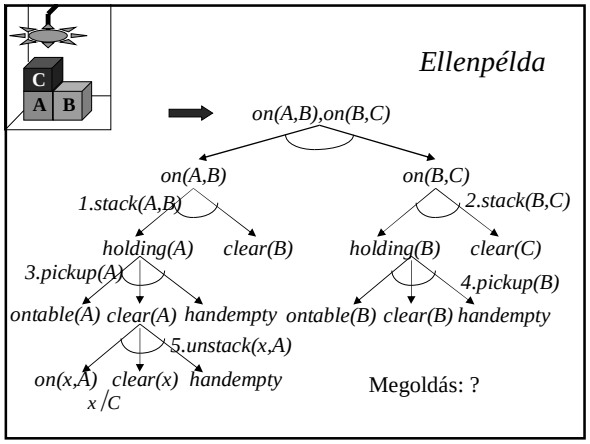
Dekompozíciós reprezentáció

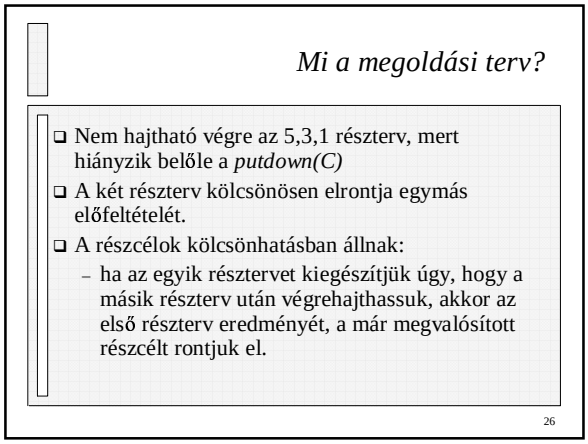
- Probléma leírás: *start→állapot*
- Kiinduló probléma: *start→cél*
- Egyszerű probléma: *start→L, ha L∈ start*
- Operátorok:
  - A *start→* $L_1 \wedge ... \wedge L_n$ visszavezethető a *start→* L_1, ... , *start→* L_n problémákra
  - A A *start→L* visszavezethető a *start→*P^M problémára, ahol M olyan művelet, hogy az $L \in A^M$

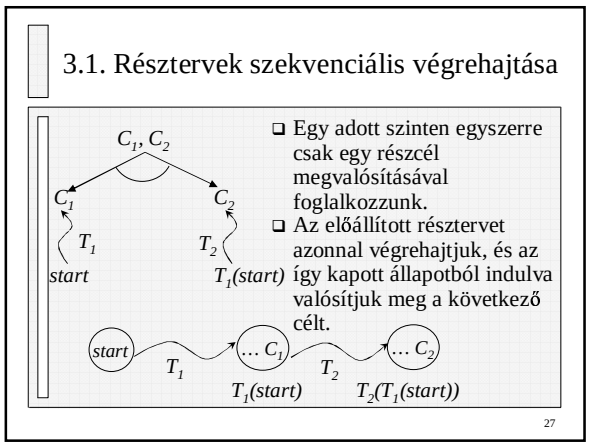
Példa

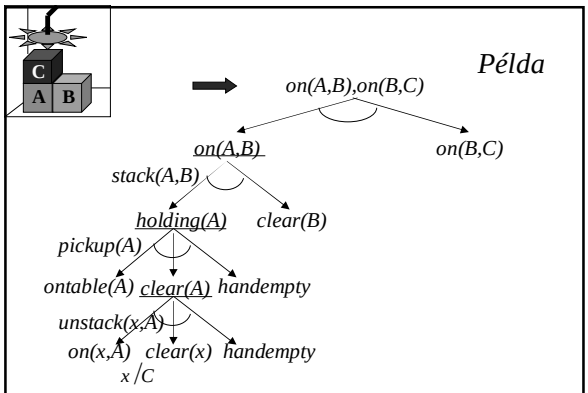


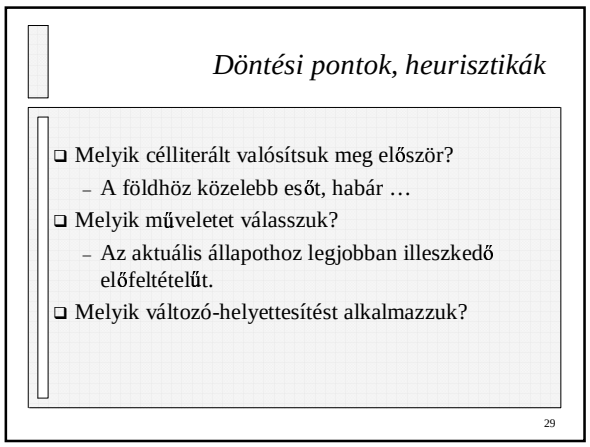
Megoldás: *3,1,4,2* vagy *4,2,3,1*

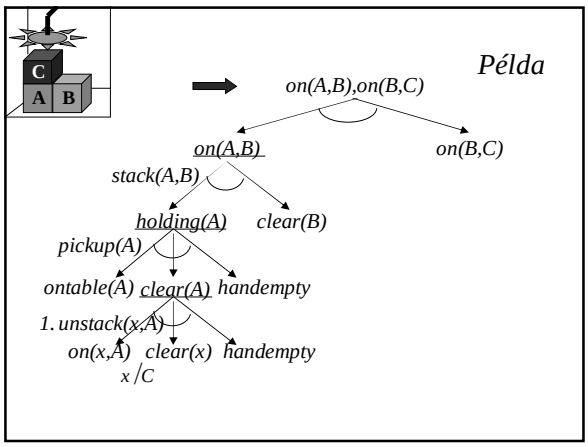

Ellenpélda
C
A B
on(A,B),on(B,C)
on(A,B)
on(B,C)
1.stack(A,B)
2.stack(B,C)
holding(A)
clear(B)
holding(B)
clear(C)
3.pickup(A)
4.pickup(B)
ontable(A) clear(A) handempty
ontable(B) clear(B) handempty
5.unstack(x,A)
on(x,A) clear(x) handempty
x /C
Megoldás: ?

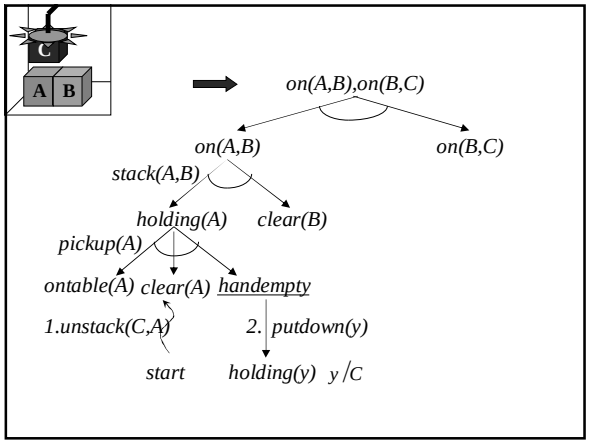

Mi a megoldási terv?
Nem hajtható végre az 5,3,1 részterv, mert hiányzik belőle a putdown(C)
A két részterv kölcsönösen elrontja egymás előfeltételét.
A részcélok kölcsönhatásban állnak:
ha az egyik résztervet kiegészítjük úgy, hogy a másik részterv után végrehajthassuk, akkor az első részterv eredményét, a már megvalósított részcélt rontjuk el.
26

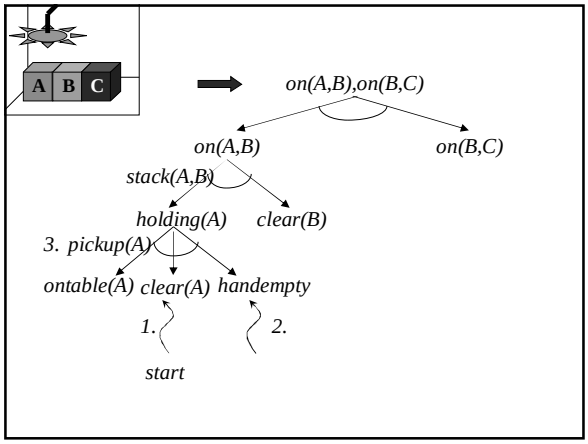

3.1. Résztervek szekvenciális végrehajtása
C1, C2
C1
C2
T1
T2
start
T1(start)
Egy adott szinten egyszerre csak egy részcél megvalósításával foglalkozzunk.
Az előállított résztervet azonnal végrehajtjuk, és az így kapott állapotból indulva valósítjuk meg a következő célt.
start
… C1
… C2
T1
T2
T1(start)
T2(T1(start))
27

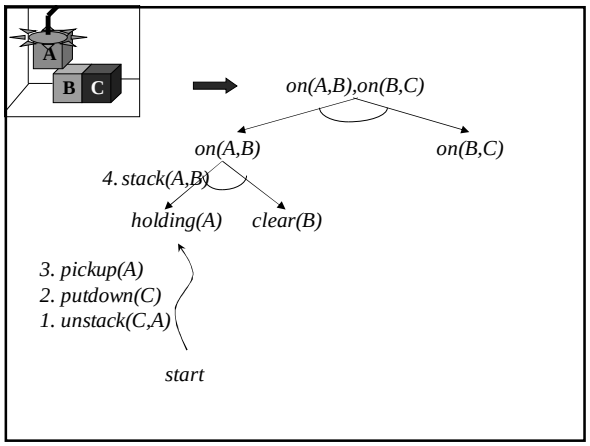

Példa
C
A B
on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
pickup(A)
ontable(A) clear(A) handempty
unstack(x,A)
on(x,A) clear(x) handempty
x /C

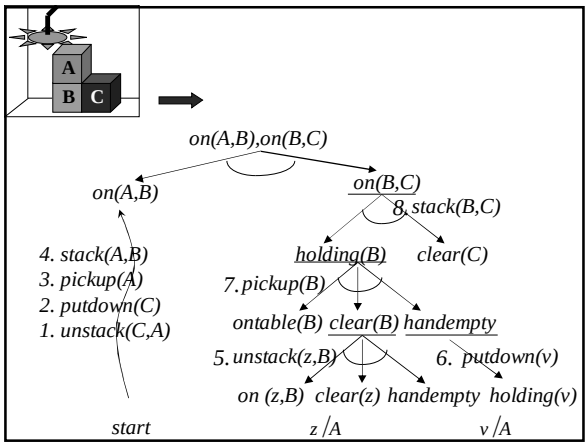

Döntési pontok, heurisztikák
Melyik célliterált valósítsuk meg először?
A földhöz közelebb esőt, habár …
Melyik műveletet válasszuk?
Az aktuális állapothoz legjobban illeszkedő előfeltételűt.
Melyik változó-helyettesítést alkalmazzuk?
29

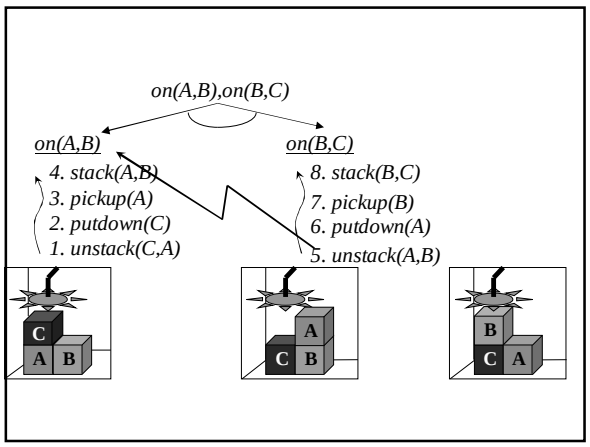

Példa
C
A B
on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
pickup(A)
ontable(A) clear(A) handempty
1. unstack(x,A)
on(x,A) clear(x) handempty
x /C

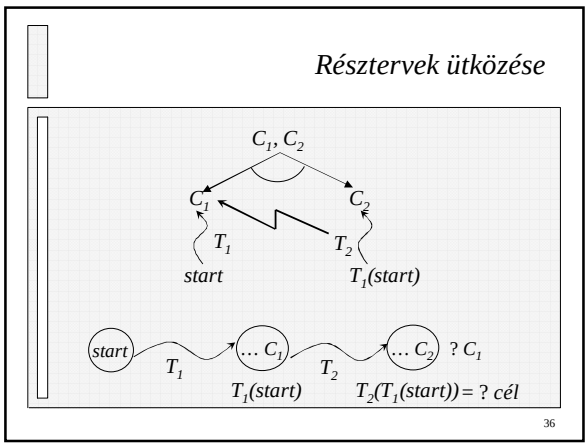

C
A B
on(A,B),on(B,C)
on(A,B)
on(B,C)
stack(A,B)
holding(A)
clear(B)
pickup(A)
ontable(A)
clear(A)
handempty
1.unstack(C,A)
2. putdown(y)
start
holding(y)
y /C


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

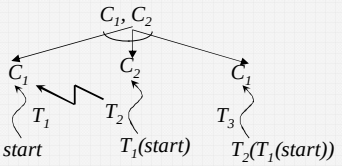

A
B C
on(A,B),on(B,C)
on(A,B)
on(B,C)
4. stack(A,B)
holding(A)
clear(B)
3. pickup(A)
2. putdown(C)
1. unstack(C,A)
start


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

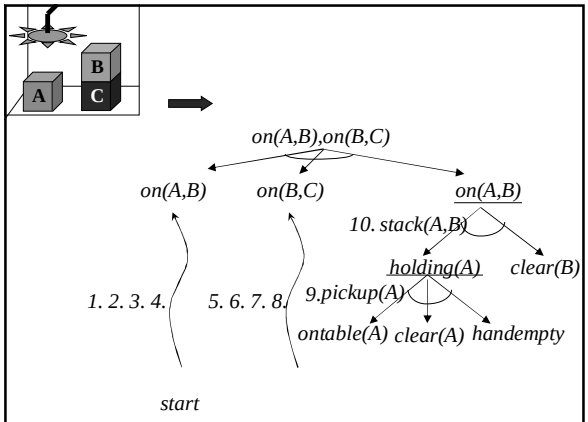

Résztervek ütközése
C1, C2
C1
C2
T1
T2
start
T1(start)
start
… C1
… C2
? C1
T1
T2
T1(start)
T2(T1(start)) = ? cél
36

I. STRIPS



Hagyjuk, hogy egy későbbi (T_2) részterv elrontson egy már megvalósított (C_1) részcélt, amit később újra megvalósítunk az aktuális állapotból kiindulva.

37

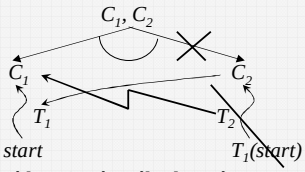


Megjegyzés

- STRIPS mindig talál megoldást, ha a műveleteknek van inverze
- Ellenpélda: Regiszter csere

39

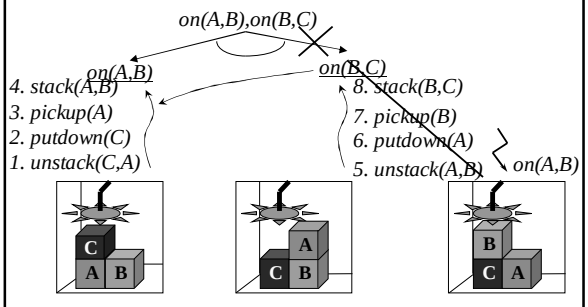
II. RSTRIPS

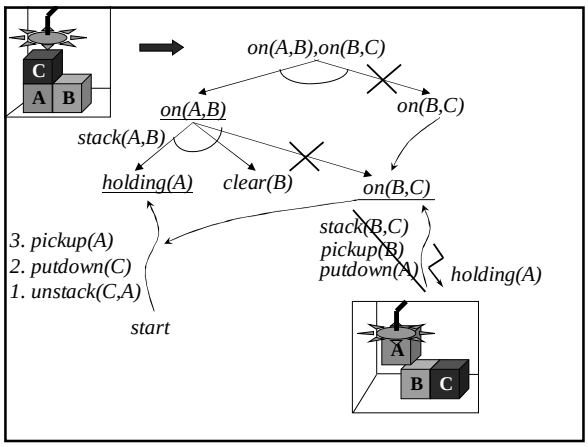


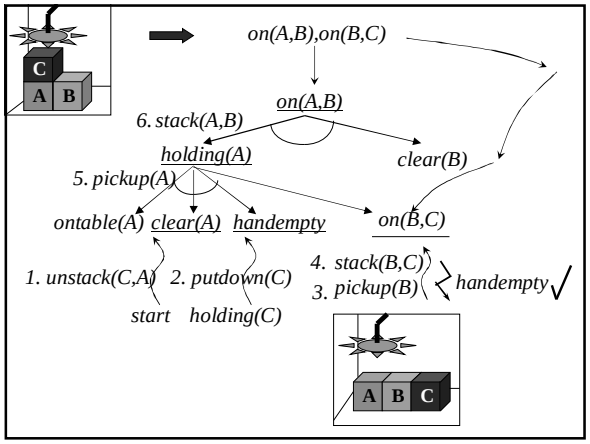
- Töröljük a védett C_1 részcélt elrontó tervet.
- Valósítsuk C_2-t a T_1 részterv utolsó művelete előtt
  - legyen ennek a műveletnek a C_2 egy újabb előfeltétele
  - feltéve, hogy C_2-t nem törli ez a művelet.

40

Példa







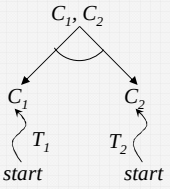
Megjegyzés

- Előrehelyezés esetei
  - Ha a literált megvalósító terv véglegesen elrontja annak már megvalósított testvér-literálját.
  - Ha a literált nem tudjuk megvalósítani. (kör, zsákutca)
- Nem helyezhető előre, ha az a művelet, ami elé akarjuk helyezni, törli a literált.
- Visszalépéses keresésbe ágyazva
- Az RSTRIPS talál megoldást, ha van megoldás?
  - Regiszter-csere

44

3.2. Résztervek összefésülése

Térjünk vissza az eredeti dekompozícióhoz, és a párhuzamosan előállított résztervek műveletei között jelöljünk ki - ha kell - végrehajtási sorrendet, sorrendi kötéseket.



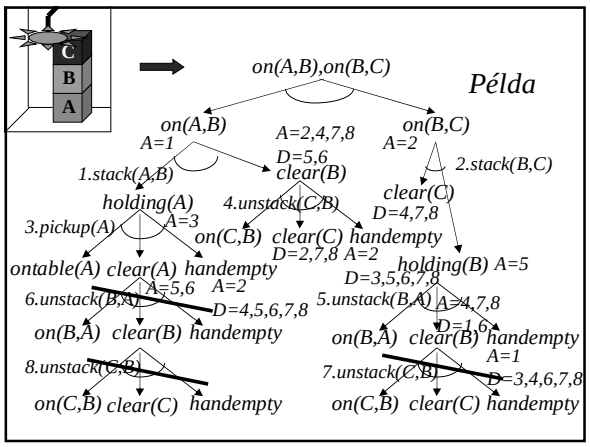
Az így kapott tervet nem kell feltétlenül szekvenciába fűzni.

45

III. DCOMP

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

46



Sorrendi megkötéseket teszünk

- 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* művelete(ke)t, amely az *i* után és a *j* előtt végrehajtva előállítja a sérült feltételt.

48

Példa

1. részterv: $4 \rightarrow 5 \rightarrow 3 \rightarrow 1$
2. részterv: $4 \rightarrow 1$
3. részterv: $4 \rightarrow 5 \rightarrow 2$

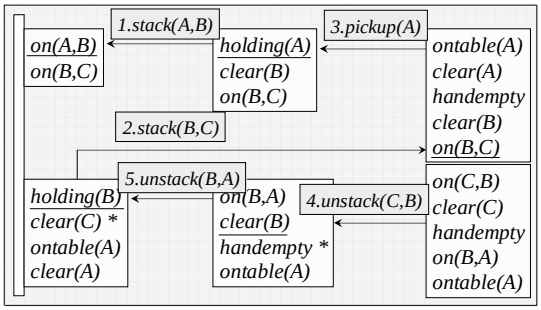
1,2:	2,3:	3,4:	4,5:
$1 \leftrightarrow 2$	$2 \leftrightarrow 3$	$4 \rightarrow 2 \rightarrow 3$	$4 \rightarrow ? \rightarrow 5$
1,5:	2,4:	3,5:	
$5 \rightarrow 2 \rightarrow 1$	$4 \rightarrow ? \rightarrow 2$	$5 \rightarrow 2 \rightarrow 3$	

$4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 1$
handempty clear(C)

Az ütközéses terv preparálása

- A cél állapot leírásából elindulva, a kijelölt műveletek mentén redukció segítségével kiszámoljuk a tervezett megoldás mentén szereplő állapotokat.
- Az ütközést okozó sérült literálokat nem regrettáljuk.
- Ezután a start állapot felől elindulva kijelöljük az első ütközést feloldó részfeladatot (mi van és minek kéne lenni), megoldjuk, és megyünk tovább.

$4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 1$
handempty clear(C)



Példa

- A 4. művelet (*unstack(C,B)*) végrehajtása után
  - *holding(C), clear(B), on(B,A), ontable(A)*
- Előállítandó részcél:
  - *on(B,A), clear(B), handempty*, ontable(A)*
- Megoldás:
  - *putdown(C)*
- A *clear(C)*-vel szerencsénk van, mert a *putdown(C)* ezt is előállította.

IV. NONLIN

- Párhuzamosítjuk a DCOMP első két fázisát:
  - Nem akarjuk a műveletek 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.

Több-robotkaros kockavilág

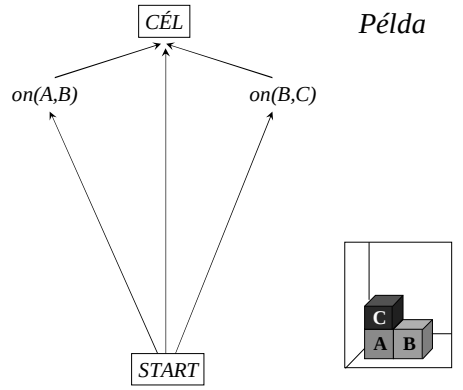
- Több-robotkaros kockavilágban a NONLIN párhuzamos terv előállítására is képes
  - Adott számú robotkar esetén ki kell egészíteni a handempty és a holding állításokat egy robotkarra utaló argumentummal.
  - Tetszőleges számú robotkar esetén töröljük a handempty állítást.
    - Ekkor a tervgenerálástól várjuk a szükséges robotkarok számának meghatározását is
- Egy robotkar esetén egy-szálú terv készül.

NONLIN algoritmusa

- Egy folyamot (irányított, körmentes !, egy nyelő (cél) és egy forrás (start) csúcsot tartalmazó gráf, ahol minden csúcson keresztül vezet legalább egy startból célba tartó út) építő algoritmus:
  - Kétféle csúcs: művelet vagy literál
  - Kétféle él: literál címkével vagy anélkül
  - A start és célcsúcs egy-egy speciális művelet.
  - Kezdetben: a start-csúcsból vezet él a célcsúcsba.
  - A gráf mindig egy ütközés mentes tervet ábrázol, amely akkor teljes, ha nincs benne literál-csúcs.

55

Példa

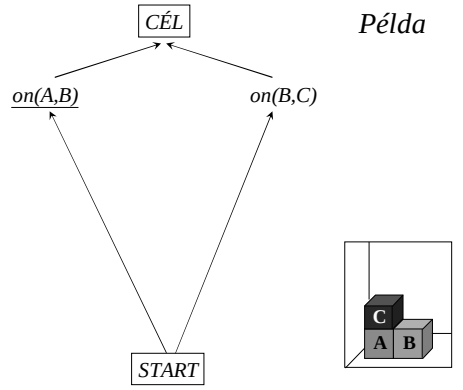


Gráf-egyszerűsítés

- Él törlése:
  - Dupla élt összevonunk.
  - Az (a,b) él törölhető, ha van ezt az élt elkerülő $a \rightarrow b$ út a gráfban.
  - A törölt él címkéjét (ha van) át kell írni az elkerülő út utolsó élére.

57

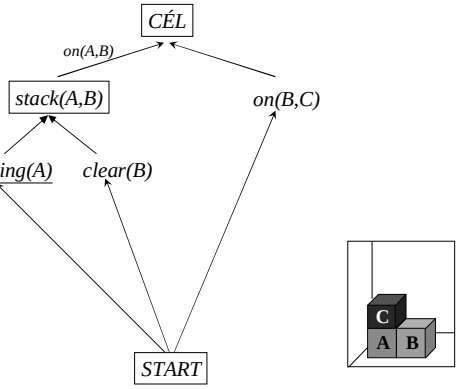
Példa

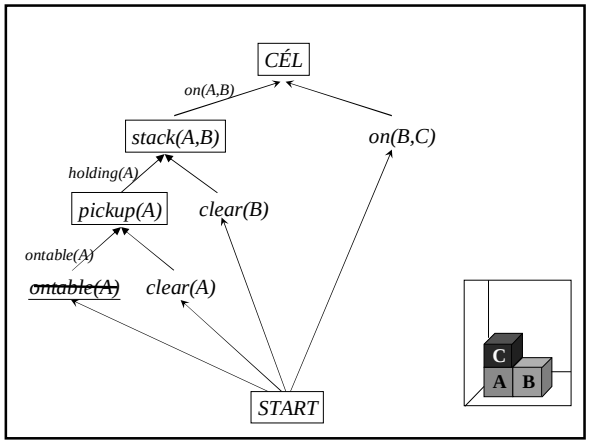


Literál-csúcs eliminálása 1.

- Ha nincs olyan művelet (nem lehet leszármazottja a literálnak, és nem lehet őse a literál szülőcsúcsának) amelyik a literált előállítja vagy van, de a művelet-csúcsból a literálba vezető úton van egy a literált kitörlő másik művelet
- akkor egy olyan új műveletet illesztünk a hálóba, amelynek bővítési listáján szerepel a literál.
  - a literál-csúcsot a művelet-csúccsal helyettesítjük
  - a műveletből kifutó él címkéje a literál lesz
  - felvesszük új csúcsokként a művelet előfeltételének literáljait, és belőlük élt húzunk a művelet-csúcshoz, és mindegyikhez élt húzunk a literál szülőcsúcsából.

59

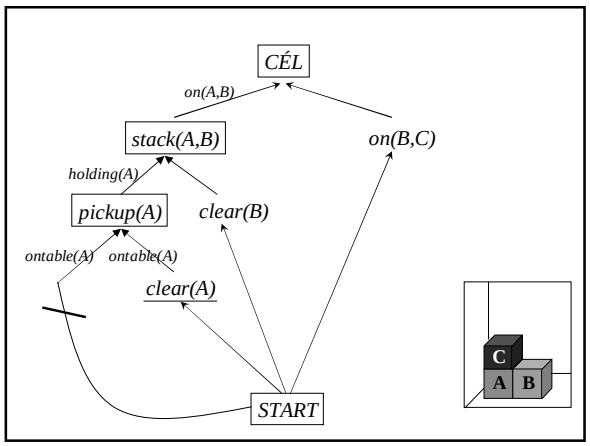


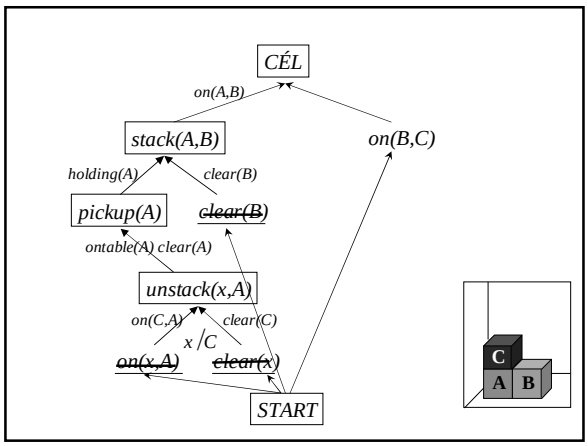


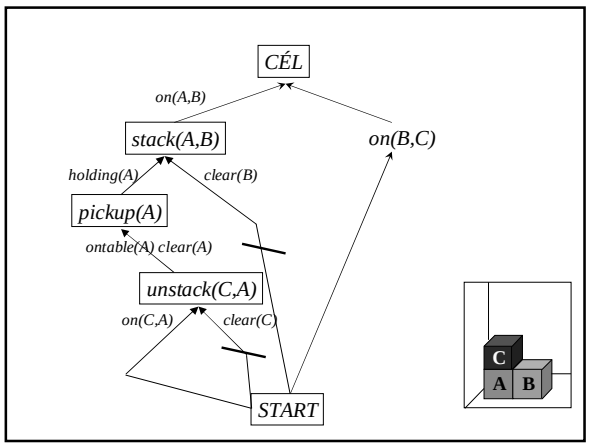
Literál-csúcs eliminálása 2.

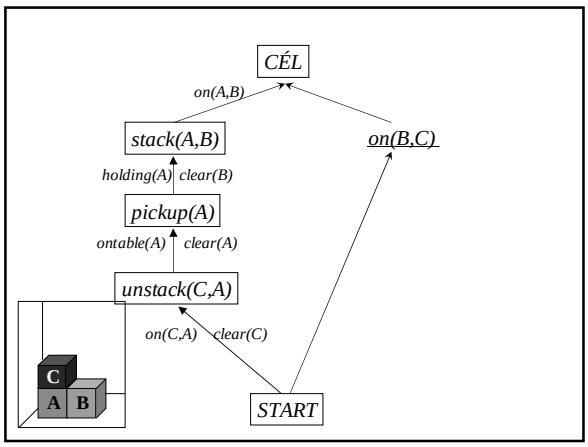
- Ha a literált előállítja egy művelet (nem leszármazottja a literálnak, és nem őse a literál szülőcsúcsának) és a művelet-csúcsból a literálba vezető úton nincs a literált kitörlő másik művelet, akkor
  - a literál-csúcsot belőle kivezető éllel töröljük,
  - a műveletből élt húzunk a literál-csúcs szülőjéhez
  - az él címkéje a literál lesz

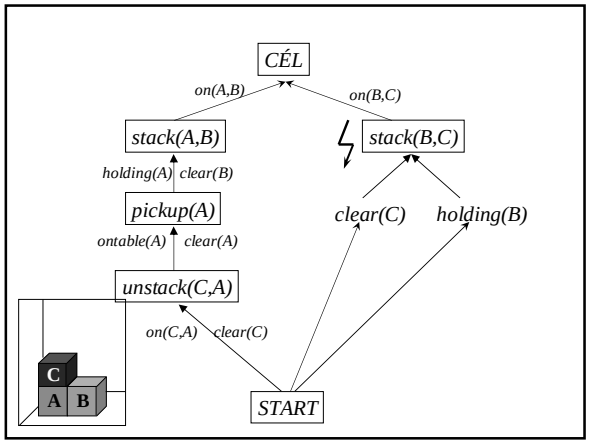
62











Ütközés (sérül a konzisztencia)

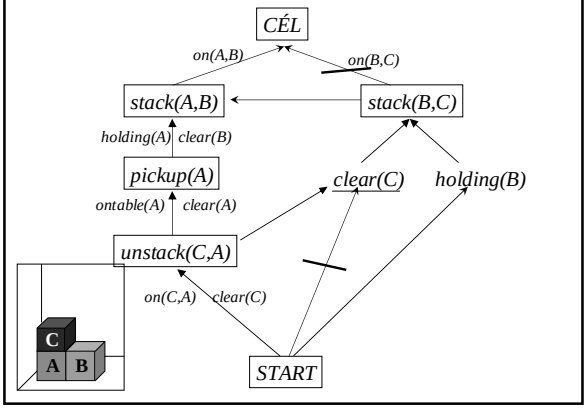
- Egy *új* művelet (amit most generáltuk vagy változó kötést kapott) "ütközhet" már meglevő *régi* művelettel:
  - Nincs köztük sorrend, de közös erőforrást használnak.
  - Van köztük kijelölt sorrend és az egyik sérti a másikat (törlési, bővítési és előfeltétel listák összevetése), azaz az előbb végrehajtandó művelet (véglegesen) törli az utóbbi egyik előfeltételét
  - Súlyos sértésről beszélünk, ha a törölt literált már korábban megvalósítottuk, azaz szerepel valamelyik él címkéjeként.

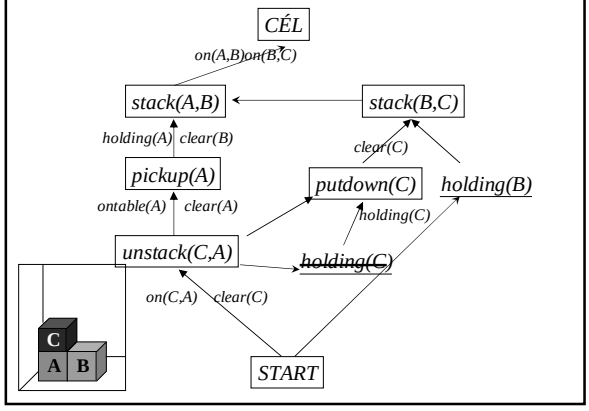
68

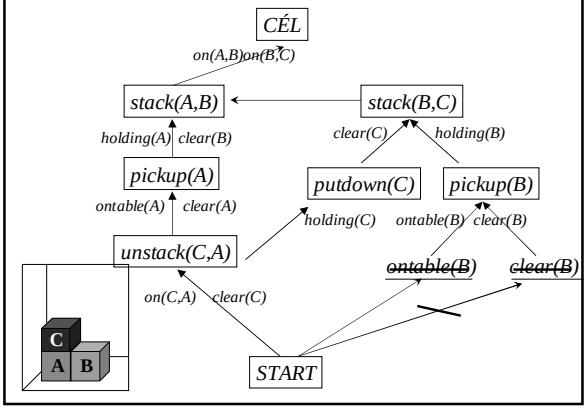
Ütközés feloldása

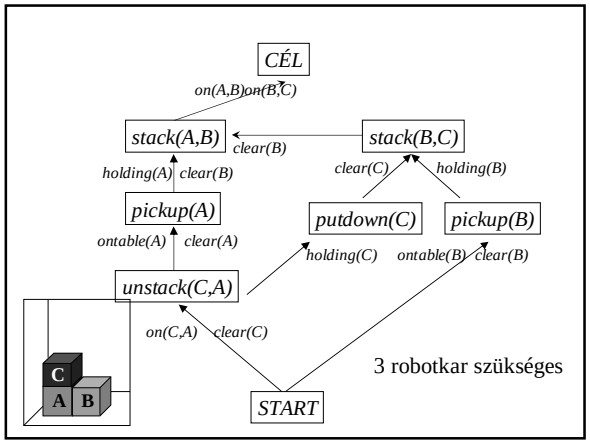
- Sorrendet (lehetőleg sértésmentes vagy nem súlyosan sértő sorrendet) jelölünk ki a két művelet között
  - Ha a sorrend sértésmentes, akkor egy irányított élt húzunk az előbb végrehajtandó műveletből a másikba. Ügyeljünk arra, hogy ne alakuljon ki irányított kör.
  - Ha a sorrend sértő, mert az előbb végrehajtandó művelet (a sértő) véglegesen törli a későbbi (a sértett) *L* előfeltétel-literálját, akkor egy irányított élt húzunk sértő műveletből az *L* literál-csúcshoz.
  - Ha ráadásul a sorrend súlyosan sértő is, akkor a fentiek mellett töröljük a sértett műveletbe befutó élek címkéi közül az *L* literált.

69









4. Logikai reprezentációk

- Logikai következtetéssel (rezolúció, szabály alapú következtetés) generáljunk tervet!
- A logikai reprezentáció nem is olyan egyszerű.

74

4.1. Naiv reprezentáció

- Kézenfekvő logikai reprezentáció:
  - Tény:
    - kezdőállapot leírása
  - Szabályok:
    - műveletekből származó $P^F \rightarrow A^F$ állítások
  - Cél:
    - célállapot leírása
- Bizonyítandó:
  kezdőállapot, műveleti szabályok $\Rightarrow$ célállapot

75

1.Példa

- Kezdőállapot:
  - *handempty, clear(A), clear(B), ontable(A), ontable(B)*
- Célállapot:
  - *on(A,B)*
- Szabályok:
  - $\forall x \forall y$ *(holding(x)* $\wedge$ *clear(y)* $\rightarrow$ *on(x,y)* $\wedge$ *clear(x)* $\wedge$ *handempty)*
  - $\forall x \forall y$ *(on(x,y)* $\wedge$ *clear(x)* $\wedge$ *handempty* $\rightarrow$ *holding(x)* $\wedge$ *clear(y))*
  - $\forall x$ *(holding(x)* $\rightarrow$ *ontable(x)* $\wedge$ *clear(x)* $\wedge$ *handempty)*
  - $\forall x$ *(ontable(x)* $\wedge$ *clear(x)* $\wedge$ *handempty* $\rightarrow$ *holding(x))*

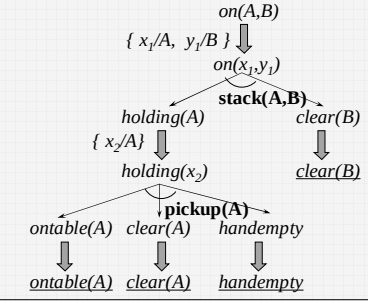
76

Visszafelé haladó szabályalapú reprezentáció

- Tény:
  - *handempty, clear(A), clear(B), ontable(A),ontable(B)*
- Szabályok:
  - $\forall x$ *(ontable(x)* $\wedge$ *clear(x)* $\wedge$ *handempty* $\rightarrow$ *holding(x))*
  - $\forall x \forall y$ *(holding(x)* $\wedge$ *clear(y)* $\rightarrow$ *on(x,y))*
  - $\forall x \forall y$ *(holding(x)* $\wedge$ *clear(y)* $\rightarrow$ *clear(x))*
  - $\forall x \forall y$ *(holding(x)* $\wedge$ *clear(y)* $\rightarrow$ *handempty)*
  - …
- Cél:
  - *on(A,B)*

77

Szabályalapú következtetés



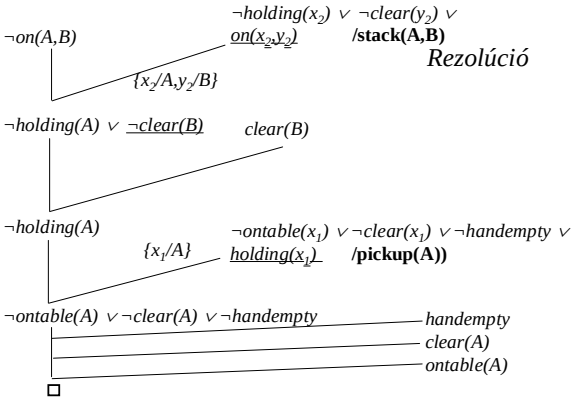
78

Klóz formára hozás

- axiómák:
  - *handempty, clear(A), clear(B), ontable(A),ontable(B)*
  - $\neg ontable(x_1) \vee \neg clear(x_1) \vee \neg handempty \vee holding(x_1)$
  - $\neg holding(x_2) \vee \neg clear(y_2) \vee on(x_2,y_2)$
  - $\neg holding(x_3) \vee \neg clear(y_3) \vee clear(x_3)$
  - $\neg holding(x_4) \vee \neg clear(y_4) \vee handempty$
  - …
- Cél:
  - $\neg on(A,B)$

79

Rezolúció



2.Példa

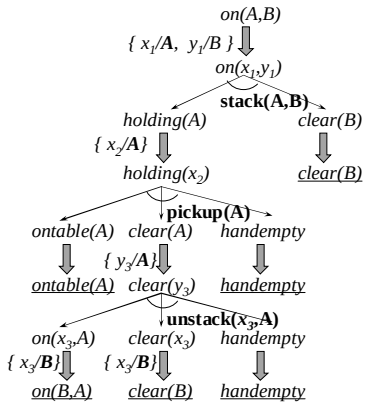
- Kezdőállapot:
  - *handempty, clear(B), on(B, A), ontable(A)*
- Célállapot:
  - *on(A,B)*
- Szabályok:
  - $\forall x \forall y\ (holding(x) \wedge clear(y) \rightarrow on(x,y) \wedge clear(x) \wedge handempty)$
  - $\forall x \forall y\ (on(x,y) \wedge clear(x) \wedge handempty \rightarrow holding(x) \wedge clear(y))$
  - $\forall x\ (holding(x) \rightarrow ontable(x) \wedge clear(x) \wedge handempty)$
  - $\forall x\ (ontable(x) \wedge clear(x) \wedge handempty \rightarrow holding(x))$

81

Visszafelé haladó szabályalapú reprezentáció

- Tény:
  - *handempty , clear(B) , on(B,A) , ontable(A)*
- Szabályok:
  - $\forall x\ (holding(x) \rightarrow ontable(x) \wedge clear(x) \wedge handempty)$
  - $\forall x \forall y\ (on(x,y) \wedge clear(x) \wedge handempty \rightarrow clear(y))$
  - $\forall x\ (ontable(x) \wedge clear(x) \wedge handempty \rightarrow holding(x))$
  - …
- Cél:
  - *on(A,B)*

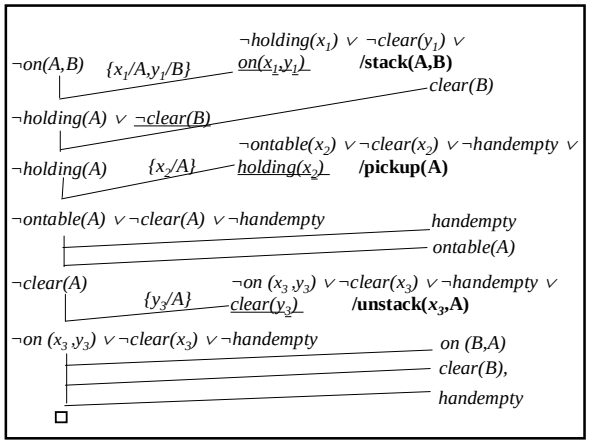
82



Klóz formára hozás

- Axiómák:
  - *handempty , clear(B) , on(B,A) , ontable(A)*
  - $\neg holding(x_1) \vee \neg clear(y_1) \vee on(x_1,y_1)$
  - $\neg on(x_3,y_3) \vee \neg clear(x_3) \vee \neg handempty \vee clear(y_3)$
  - $\neg ontable(x_2) \vee \neg clear(x_2) \vee \neg handempty \vee holding(x_2)$
  - …
- Cél:
  - $\neg on(A,B)$

84



Problémák

- Nem generál tervet a válaszadási eljárás:
  - Az érvényesé kiegészített *on(A,B)* ∨ ¬*on(A,B)* célklózban nem tud megjelenni a válasz.
  - Abból, hogy mely művelet axiómájával rezolváltunk, még nem nem olvasható ki az alkalmazott műveletek sorrendje.
  - Az üres klóz levezetéséhez nem használtuk a *putdown* művelet axiómáit, pedig arra biztos szükség van.

86

4.2. Tervgenerálás

- Logikai reprezentáció készítése állapottér-reprezentációból: szituáció kalkulus.
- Ez nem azonos a terv létezésének eldöntésénél használt reprezentációval.

87

Predikátum szimbólumok

- A $P(\underline{x})$ állítást egy állapot jelzésére alkalmas változóval kell kiegészíteni. A $P(\underline{x},s)$ predikátum így mindig az adott *s* állapotra vonatkozik majd.
- A kocka-világ probléma predikátumai:
  - *ontable(x,s), on(x,y,s), clear(x,s), holding(x,s), handempty(s)*
  - *Például az* s_0 kezdőállapotban
    - $ontable(A,s_0)$, $ontable(B,s_0)$, $clear(A,s_0)$, $clear(B,s_0)$, $handempty(s_0)$

88

Függvény szimbólumok

- Az $F(\underline{x})$ művelet egy *állapot→állapot* operátor, amit *paraméterek×állapot→állapot* függvényszimbólumnak tekinthetünk.
  - Az $F(\underline{x})(s)$ vagy röviden $F(\underline{x},s)$ megadja azt az állapotot amelyikbe az *s*-ből az $F(\underline{x})$ művelet végrehajtásával jutunk.
- A kocka-világ probléma függvényei:
  - *pickup(x,s), putdown(x,s), stack(x,y,s), unstack(x,y,s)*
- Egy terv ábrázolása:
  - Pl: $stack(A,B,pickup(A,s_0))$

89

Alternatív megoldás az állapot átmenetek jelölésére

- Az $F(\underline{x})$ műveletek függvényszimbólumai mellett bevezetjük a *do:művelet×állapot→állapot* operátort, amit szintén függvényszimbólumnak tekintünk.
  - A $do(F(\underline{x}),s)$ megadja azt az állapotot amelyikbe az *s*-ből az $F(\underline{x})$ művelet végrehajtásával jutunk.
- Egy terv ábrázolása:
  - Pl: $do(stack(A,B),do(pickup(A),s_0))$

90

Tény-állítások

- Kezdőállás predikátumai az s_0 kezdőállapotban
  - Pl: $ontable(A,s_0)$, $ontable(B,s_0)$, $clear(A,s_0)$, $clear(B,s_0)$, $handempty(s_0)$
- Egyéb megkötések
  - $A \neq B$, $B \neq A$

91

Hatás-szabályok

- A *pickup(x)* művelet a *P* listáján szereplő állítások teljesülése esetén hajtható végre, és végrehajtása után teljesülni fognak az *A* listáján szereplő állítások.
  - $\forall x \forall s\ ontable(x,s), clear(x,s), handempty(s) \rightarrow holding(x, pickup(x,s))$
- Általában az $F(\underline{x})(P,A,D)$ művelet hatás-szabálya a
  - $\forall \underline{x} \forall s\ P^F(\underline{x},s) \rightarrow A^F(\underline{x}, F(\underline{x},s))$

92

Keret-probléma

- Tegyük fel $ontable(A,s)$, $ontable(B,s)$, $clear(A,s)$, $clear(B,s)$, $handempty(s)$.
- Alkalmazzuk a *pickup(A)* műveletet az *s* állapotban. Így a *pickup(A,s)* állapotba jutunk.
- A művelet hatás-szabályából kiolvasható, hogy az új állapotban: $holding(A, pickup(A,s))$
- De hogyan adjuk meg azt, hogy az új állapotban $ontable(B, pickup(A,s))$, és a $clear(B, pickup(A,s))$ állítás is teljesül?
- Vegyük észre, hogy ezek éppen azok, amelyek <u>nem</u> szerepeltek a *pickup(A) D* listáján.

93

Keret-probléma megoldása

- Keret szabályokkal:
  - Explicit módon leírjuk minden műveletre azt, hogy azok végrehajtásakor mely állítások mely állítások maradnak meg az új állapotban
  - Pl: $ontable(B,s) \rightarrow ontable(B, pickup(A,s))$

    $clear(B, s) \rightarrow clear(B, pickup(A,s))$
  - Vegyük észre, hogy ezek éppen azok az állítások, amelyek <u>nem</u> szerepeltek a *művelet D* listáján.
- Default szabályokkal: $\dfrac{Q(\underline{y},s) : Q(\underline{y}) \notin D^{F(x)}}{Q(\underline{y}, F(\underline{x}, s))}$

94

Keret-szabályok

- A *pickup(x)* művelet a *D* listáján <u>nem</u> szereplő állítások a $clear(u,s)$ (ahol $u \neq x$), az $ontable(u,s)$ (ahol $u \neq x$), és az $on(u,v,s)$. Ezért
  - $\forall x \forall u \forall s\ clear(u,s)\ , u \neq x \rightarrow clear(u,pickup(x,s))$
  - $\forall x \forall u \forall s\ ontable(u,s)\ , u \neq x \rightarrow ontable(u,pickup(x,s))$
  - $\forall x \forall u \forall v \forall s\ on(u,v,s) \rightarrow on(u,v,pickup(x,s))$
- Általában az $F(\underline{x})(P,A,D)$ művelet keret-szabálya az $L(\underline{u})$ literálra :
  - $\forall \underline{x} \forall \underline{u} \forall s\ L(\underline{u},s) \wedge L(\underline{u}) \notin D^F(\underline{x}) \rightarrow L(\underline{u}, F(\underline{x},s))$

95

Összefoglalva

- Tények
  - kezdő állapot
  - egyéb állítások
- Szabályok:
  - hatás-szabályok:
    $\forall s \forall \underline{x}\ P^F(\underline{x},s) \rightarrow A^F(\underline{x}, F(\underline{x},s))$
  - keret-szabályok:
    $\forall s \forall \underline{x} \forall \underline{u}\ L(\underline{u},s) \wedge L(\underline{u}) \notin D^F(\underline{x}) \rightarrow L(\underline{u},F(\underline{x},s))$
- Cél:
  $\exists t\ cél(t)$

96

Példa

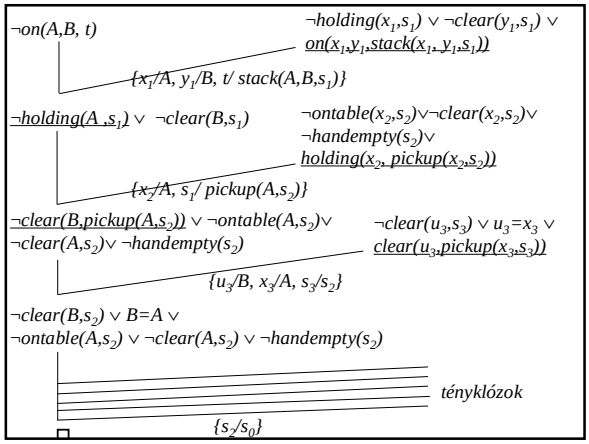
- Tények: $ontable(A,s_0)$, $ontable(B,s_0)$, $clear(A,s_0)$, $clear(B,s_0)$, $handempty(s_0)$, $A \neq B$, $B \neq A$
- Hatás-szabályok: (most csak kettő)
  - $\forall x \forall s\ ontable(x,s), clear(x,s), handempty(s) \rightarrow holding(x, pickup(x,s))$
  - $\forall x \forall y \forall s\ holding(x,s), clear(y,s) \rightarrow on(x,y,stack(x,y,s)), clear(y, stack(x,y,s)), handempty(stack(x,y,s))$
- Keret-szabályok: (most csak egy)
  - $\forall x \forall u \forall s\ clear(u,s)\ ,\ u \neq x \rightarrow clear(u,pickup(x,s))$
- Cél: $\exists t\ on(A,B,t)$

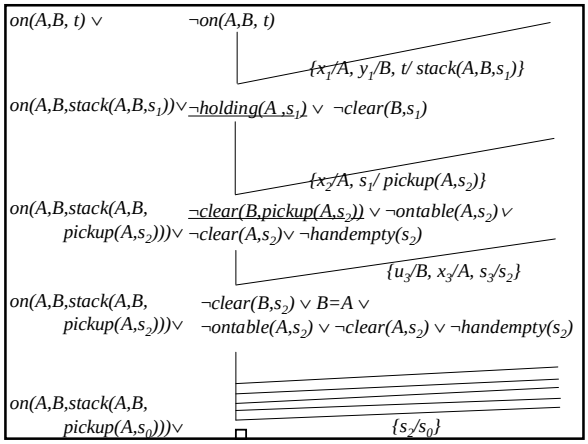
97

Rezolúció

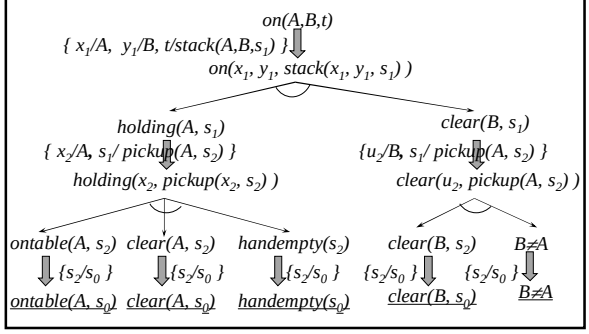
- $ontable(A,s_0)$, $ontable(B,s_0)$, $clear(A,s_0)$, $clear(B,s_0)$, $handempty(s_0)$, $A \neq B$, $B \neq A$
- $\neg ontable(x,s) \vee \neg clear(x,s) \vee \neg handempty(s) \vee holding(x, pickup(x,s))$
- $\neg holding(x,s) \vee \neg clear(y,s) \vee on(x,y,stack(x,y,s))$
- $\neg holding(x,s) \vee \neg clear(y,s) \vee clear(y, stack(x,y,s))$
- $\neg holding(x,s) \vee \neg clear(y,s) \vee handempty(stack(x,y,s))$
- $\neg clear(u,s) \vee u=x \vee clear(u,pickup(x,s))$
- $\neg on(A,B,t)$

98





Szabályalapú következtetés



Megjegyzés

- Jól jönne egy kis heurisztika.
  - Egy adott szinten ugyanazon művelet szabályait illesszük
- Másik állapottér-reprezentáció segíthet (Green).
- Műveleti (hatás és keret) szabályok száma általában csökkenthető Kowalski módszerével.

102

Kowalski reprezentációja

Túl sok szabály van, mert

- egy hatás-szabály több $W \rightarrow L$ szabályra esik szét
- egy műveletnek sok keret-szabálya van
  - $\forall F \forall L \forall s \forall \underline{x} \forall \underline{u}\ (L(\underline{u},s) \wedge L(\underline{u}) \notin D^F(\underline{x}) \rightarrow L(\underline{u},F(\underline{x},s)))$

    (másodrendű logikával kellene dolgoznunk)

103

Ötlet

- Univerzum
  - kockák, állapotok, állítások, műveletek
- Predikátum szimbólumok:
  - *HOLD(állítás,állapot)*
  - *PACT(művelet,állapot)*
  - *POSS(állapot)*
- Függvény szimbólum:
  - *do(művelet,állapot)*

104

Tényállítás

- Hatás-szabályok:
  - $HOLD(a,do(F(\underline{x}),s))$ ha $a \in A^F(\underline{x})$
    - $HOLD(holding(x),do(pickup(x),s))$
    - $HOLD(on(x,y),do(stack(x,y),s))$
    - $HOLD(clear(x),do(stack(x,y),s))$
    - ...
- Kezdőállás: $HOLD(kezdet,s_0)$
  - $HOLD(ontable(A),s_0), \ldots$
- A kezdőállapot lehetséges: $POSS(s_0)$
- Speciális nem-egyenlőségek

105

Szabályok

- Előfeltételek:
  - $HOLD(P^F(\underline{x}),s) \rightarrow PACT(F(\underline{x}),s)$
    - $\forall s \forall x \forall y\ (HOLD(holding(x),s) \wedge HOLD(clear(y),s) \rightarrow PACT(stack(x,y),s)))$
- Keret-szabályok:
  - $HOLD(a,s) \wedge a \notin D^F(\underline{x}) \rightarrow HOLD(a,do(F(\underline{x}),s))$
    - $\forall s \forall a \forall x \forall y\ (HOLD(a,s) \wedge a \neq holding(x) \wedge a \neq clear(y) \rightarrow HOLD(a,do(stack(x,y),s)))$
- Lépés-szabály:
  - $POSS(s) \wedge PACT(m,s) \rightarrow POSS(do(m,s))$

106

Célállítás

- A célállítás fenn áll a célállapotban és a célállapot lehetséges
  - $\exists t\ (POSS(t) \wedge HOLD(cél,t))$
  - Példa: $\exists t\ (POSS(t) \wedge HOLD(on(A,B),t))$

107