

תרגילים בתכנון

שאלה 1

נתונה בעיית התכנון הבאה:

נתון שולחן T , מנוף C , ו- n קוביות.

כל קוביה יכולה להיות על פני השולחן, בכף המנוף, או על גבי קוביה אחרת.

כף המנוף יכולה:

- להרים (pick) קוביה מהשולחן (אם אינה אוחזת בקוביה כלשהי)
- להניח (drop) קוביה על השולחן (אם היא ת בקוביה זו)
- לערום (stack) קוביה x מעל קוביה y (אם היא אוחזת בקוביה x ואין קוביה מעל y)
- להסיר בעזרת כף המנוף (unstack) קוביה x מקוביה y (אם אינה אוחזת בקוביה x ואין קוביה מעל y).

בתחילה הקוביות $1..n$ מסודרות במגדל זו על גבי זו ונרצה להחליף את 2 הקוביות התחתונות (n ו- $n-1$).

ייצגו את הבעיה בשפת PDDL בדומה לדוגמאות שבסעיף 10.1 בספר הלימוד ובמדריך הלמידה.

פתרון שאלה 1

אוביקטים: Table והקוביות המסומנות כ-1..n.

פרדיקטים:

On(x, y) - is x on y?

Block(x) – Is x a block?

Holding(x) – Is the robot holding X?

Handempty – true is the robot is holding nothing.

Clear(x) – No block on top of the block x.

מצב התחלתי:

Init(On(n, Table) AND On(n-1, n) AND On(n-2, n-1) AND On(n-3, n-2) AND ... AND On(2, 3) AND On(1, 2))

מצב המטרה מסופי:

Goal(On(n-1, Table) AND On(n, n-1) AND On(n-2, n) AND On(n-3, n-2) AND ... AND On(2, 3) AND On(1, 2))

פעולות:

Stack(x, y): // Drop block x on top of block y

Preconditions: Holding(x) AND Clear(x) AND Clear(y) AND Block(x) AND Block(y) AND $x \neq y$

Effect: On(x, y) AND Handempty AND NOT(Holding(x)) AND NOT(Clear(y))

UnStack(x, y): // take block x, which is on block y.

Preconditions: On(x, y) AND Clear(x) AND Handempty AND Block(x) AND Block(y) AND $x \neq y$

Effect: Holding(x) AND Clear(y) AND NOT(Handempty) AND NOT(On(x, y))

Pick(x): // Take block x, which is on the table.

Preconditions: On(x, Table) AND Clear(x) AND Handempty AND Block(x)

Effect: Holding(x) AND NOT(On(x, Table)) AND NOT(Handempty)

Drop(x): // Drop block x on the table.

Preconditions: Holding(x) AND Clear(x) AND Block(x) AND NOT(Handempty)

Effect: On(x, Table) AND Handempty AND NOT(Holding(x))

שאלה 2

נתייחס לבעיית מגדלי האנוי עם 3 עמודים: p_1, p_2, p_3 ו-3 דיסקיות בגדלים שונים.

דיסקית נמצאת על עמוד: $At(disk, p)$

ויכולה להיות מעל דיסקית אחרת: $On(diskA, diskB)$

הפרדיקט $Larger(diskA, diskB)$ מקבל ערך אמת אם הדיסקית $diskA$ גדולה מ- $diskB$.

הפעולות (אופרטורים) האפשריות הן:

אם אין דיסקית מעל $diskA$

אזי

ניתן להעביר את $diskA$ כך שתהיה מעל $diskB$ בתנאי ש- $Larger(diskB, diskA)$

או

ניתן להעבירה לעמוד ריק.

א. כתבו ב-PDDL את סכמת הפעולות האפשריות כמתואר לעיל.

ב. נתייחס כעת ל-3 דיסקיות: $D1, D2, D3$ ושלושה עמודים p_1, p_2, p_3 ולמצב ההתחלתי הבא:

$Init(Larger(D3, D2) \wedge Larger(D3, D1) \wedge Larger(D2, D1) \wedge At(D1, p1) \wedge$
 $At(D2, p1) \wedge On(D1, D2) \wedge At(D3, p2))$

ולמצב המטרה:

$Goal(At(D1, p3) \wedge At(D2, p3) \wedge At(D3, p3))$

השתמשו בפעולה שהגדרתם בסעיף א' כדי לכתוב את התכנית להשגת המטרה הנתונה מתוך

המצב ההתחלתי.

פתרון שאלה 2

בשאלה מוגדרים הפרדיקטים הבאים

$At(x,p)$ – מציין שדיסקית x נמצאת על עמוד p .

$On(x,y)$ – מציין שדיסקית x נמצאת מעל דיסקית y .

$Larger(x,y)$ – מציין שדיסקית x גדולה יותר מדיסקית y .

למען הקריאות, נדמיין כי קיימות ההתניות $disk(x), pole(x)$ בכל המקומות הצפויים. נשתמש

בפרדיקטים הקיימים כדי להגדיר פרדיקטים חדשים

$EmptyPole(p)$ =

$at(x_1, p_1) \wedge at(x_2, p_2) \wedge at(x_3, p_3) \wedge (x_1 \neq x_2) \wedge (x_2 \neq x_3) \wedge (x_1 \neq x_3) \wedge (p_1 \neq p) \wedge (p_2 \neq p) \wedge (p_3 \neq p)$

$Clear(x) = \neg on(y, x) \wedge \neg on(z, x) \wedge (x \neq y) \wedge (x \neq z) \wedge (y \neq z)$

על ארבע אפשרויות - move נשתמש בכל קבוצת הפרדיקטים כדי להגדיר את פעולת ה

Move(x) – case 1 – on another disk, move to empty pole

precond – $clear(x) \wedge on(x, y) \wedge at(x, p_1) \wedge emptyPole(p_2) \wedge (p_1 \neq p_2)$

Effect : $\neg On(x, y) \wedge \neg At(x, p_1) \wedge At(x, p_2)$

Move(x) – case 2 – not on another disk, move to empty pole

precond

$clear(x) \wedge \neg on(x, y) \wedge \neg on(x, z) \wedge at(x, p_1) \wedge emptyPole(p_1) \wedge (x \neq y) \wedge (x \neq z) \wedge (y \neq z) \wedge (p_1 \neq p_2)$

Effect : $\neg At(x, p_1) \wedge At(x, p_2)$

Move(x) – case 3 – on another disk, move on a larger disk

precond

$clear(x) \wedge clear(z) \wedge on(x, y) \wedge at(x, p_1) \wedge at(z, p_2) \wedge larger(z, x) \wedge (p_1 \neq p_2) \wedge (x \neq y) \wedge (x \neq z) \wedge (y \neq z)$

Effect : $\neg On(x, y) \wedge On(x, z) \wedge \neg At(x, p_1) \wedge At(x, p_2)$

Move(x) – case 4 – not on another disk, move on a larger disk

precond

$clear(x) \wedge clear(z) \wedge \neg on(x, y) \wedge \neg on(x, z) \wedge at(x, p_1) \wedge at(z, p_2) \wedge larger(z, x) \wedge (x \neq y) \wedge (x \neq z) \wedge (y \neq z) \wedge (p_1 \neq p_2)$

Effect : $\neg At(x, p_1) \wedge At(x, p_2) \wedge On(x, z)$

סעיף ב' – נשתמש בפעולות *move* השונות שהגדרנו על מנת לכתוב את התכנית להשגת המטרה -

init:

$Larger(d_3, d_2) \wedge Larger(d_3, d_1) \wedge Larger(d_2, d_1) \wedge at(d_1, p_1) \wedge at(d_2, p_1) \wedge at(d_3, p_2) \wedge on(d_1, d_2)$

→ Move 2 (d3)

$Larger(...) \wedge at(d_1, p_1) \wedge at(d_2, p_1) \wedge at(d_3, p_3) \wedge on(d_1, d_2)$

→ Move 1 (d1)

$Larger(...) \wedge at(d_1, p_2) \wedge at(d_2, p_1) \wedge at(d_3, p_3)$

→ Move 4 (d2)

$Larger(...) \wedge at(d_1, p_2) \wedge at(d_2, p_3) \wedge at(d_3, p_3) \wedge on(d_2, d_3)$

→ Move 4 (d1)

$Larger(...) \wedge at(d_1, p_3) \wedge at(d_2, p_3) \wedge at(d_3, p_3) \wedge on(d_2, d_3) \wedge on(d_2, d_3)$

עוד פיתרון שאלה 2

א. עוד 3 פרדיקטים שנשתמש בהם, הדומים לאלו שבשאלה 1:

Clear(x) – Is there nothing on top of block\peg x?

Block(x) – Is x a block?

Peg(x) – Is x a peg?

כמו-כן, נגדיר עוד שני פרדיקטים, בשביל לחסוך את הסרבול שנגרם כתוצאה מההפרדה בין הדיסקים לעמודים (אגב, גם בשאלה הקודמת נוצר הסרבול הזה, בשל ההבדלה בין השולחן לקוביות – אבל שם הגדירו לנו את הפעולות שצריך לממש מראש והפכו את ההפרדה הזו להכרחית):

$OnTop(x, y) = (Disk(x) \wedge Peg(y) \wedge At(x, y)) \vee (Disk(x) \wedge Disk(y) \wedge On(x, y))$

$TrulyLarger(x, y) = (Disk(y) \wedge Peg(x)) \vee (Disk(x) \wedge Disk(y) \wedge Larger(x, y))$

כעת, נגדיר את הפעולה הנדרשת (שהופכת לפשוטה, בזכות הפרדיקטים הנוספים שהגדרנו):

Move(disk, from, to):

Preconditions: $TrulyLarger(to, disc) \wedge OnTop(disk, from) \wedge Clear(disk) \wedge$

$Clear(to) \wedge disc \neq from \wedge disc \neq to \wedge from \neq to$

Effect: Clear(from) AND OnTop(disc, to) AND NOT(OnTop(disc, from)) AND NOT(Clear(to))

ב. תכנית שתפתור את הבעיה הנה:

Move(D3, p2, p3)

Move(D1, D2, p2)

Move(D2, p1, D3)

Move(D1, p2, D2)

שאלה 3

נתייחס לבעיית התכנון הבאה:

נתון רכב בירושלים (J) ורוצים להגיע בנסיעה בו לים המלח (DS).

כדי שניתן יהיה לנהוג ברכב, צריך להיות מפתח במפסק ההתנעה/הצתה (switch).

נתונים 4 אופרטורים:

- Drive(J) – נהג (סע) לירושלים
- Drive(DS) – נהג (סע) לים המלח
- Insert(Key) – הכנס את המפתח למפסק ההתנעה/הצתה.
- Remove(Key) – הוצא את המפתח ממפסק ההתנעה/הצתה.

בנוסף לכך נתונים מספר prepositions לתיאור תכונות/מאפייני הבעיה:

- InPocket(Key)
- InSwitch (Key)
- At(Car, J)
- At(Car, Ds)

במצב ההתחלתי, המפתח בידנו ונרצה שיהיה לנו את המפתח גם בסיום התכנית.

א. תארו את 4 האופרטורים ב-PDDL.

ב. בנו את גרף התכנון ואת התכנית המוחלשת (Relaxed Plan).

מהן ההערכות היוריסטיות של המצב ההתחלתי?

פיתרון שאלה 3

נשים לב כי נוכל להסיק כי $at(Car, DS) = \neg at(Car, DS)$ וכי $InSwitch(Key) = \neg inPocket(key)$ ולכן נוכל לצמצם את מספר הפרדיקטים שלנו בחצי.

נגדיר את ארבעת האופרטורים -

Drive (J) -

$\neg at(Car, J) \wedge InSwitch(Key)$ Precond:

$at(Car, J)$ Effect:

Drive (DS) -

$at(Car, J) \wedge InSwitch(Key)$ Precond:

$\neg at(Car, J)$ Effect:

Insert (Key) -

$inPocket(key)$ Precond:

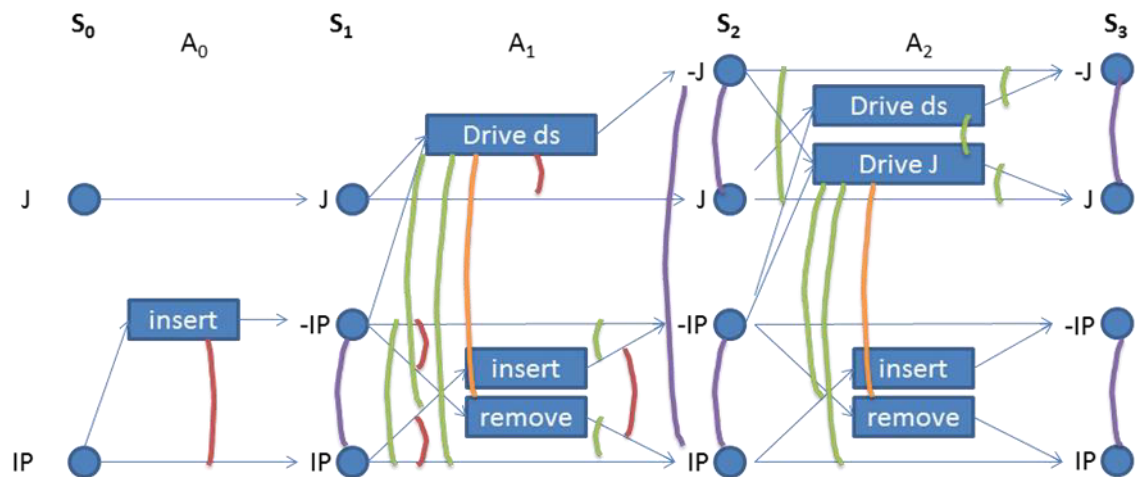
$\neg inPocket(key)$ Effect:

Remove (Key) -

$\neg inPocket(key)$ Precond:

$inPocket(key)$ Effect:

מצורף גרף התכנון. בין מצב A_1 , A_2 לא חזרתי על כל הקשתות הקיימות וכולן נותרו כפי שהיו ב A_7 . הוספה שלהם פשוט הופכת את השרטוט להיות בלתי קריא... צוינו רק הקשתות החדשות. כמו כן חשוב לציין שבחירת צבע הקשתות לציון סיבת ה $mutex$ היא לעיתים אקראית מכיוון וקיימת יותר מסיבה אחת למה שתי פעולות לא יכולות להתקיים בפעימה אחת.



מקרא הצבעים לקשתות mutex:

Inconsistent effect
interference
Competing needs
Inconsistent support

מהגרף עולה בצורה חד משמעית שהפתרון האידיאלי הוא $insert \rightarrow drive DS \rightarrow remove$

הקלה אפשרית לתוכנית תהא חיפוש מסלול מירושלים לאילת שאינו מתייחס למיקומו של המפתח.

הקלה נוספת היא למעשה מה שבצעתי בתרגיל זה בו הנחתי שקיימים רק שני מקומות בעולם – ירושלים וים המלח, ואם אתה לא באחד הריי שאתה בשני. בדוגמא הקלתי וקבעתי כי למפתח יש בדיוק שני מקומות אפשריים – הכיס, והסוויץ'.

היוריסטיקה בה השתמשתי לפתרון הבעיה היא $Set Level$. הפסקתי ברגע בו כל תנאי המטרה יכולו להתקיים בו זמנית.

א. לפי תיאור הבעיה, הנחתי שאחד ה-prepositions המופיע פעמיים, $\text{InPocket}(\text{Key})$, הוא בעצם $\text{InSwitch}(\text{Key})$.
 "מימוש" 4 האופרטורים:

$\text{Drive}(\text{J})$: // Drive to Jerusalem
 Preconditions: $\text{At}(\text{Car}, \text{DS}) \text{ AND } \text{NOT}(\text{At}(\text{Car}, \text{J})) \text{ AND } \text{InSwitch}(\text{Key}) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{NOT}(\text{At}(\text{Car}, \text{DS})) \text{ AND } \text{At}(\text{Car}, \text{J})$

$\text{Drive}(\text{DS})$: // Drive to the dead-sea
 Preconditions: $\text{NOT}(\text{At}(\text{Car}, \text{DS})) \text{ AND } \text{At}(\text{Car}, \text{J}) \text{ AND } \text{InSwitch}(\text{Key}) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{At}(\text{Car}, \text{DS}) \text{ AND } \text{NOT}(\text{At}(\text{Car}, \text{J}))$

$\text{Insert}(\text{Key})$: // Insert Key to switch
 Preconditions: $\text{NOT}(\text{InSwitch}(\text{Key})) \text{ AND } \text{InPocket}(\text{Key})$
 Effect: $\text{InSwitch}(\text{Key}) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$

$\text{Remove}(\text{Key})$: // Remove Key from switch
 Preconditions: $\text{InSwitch}(\text{Key}) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{NOT}(\text{InSwitch}(\text{Key})) \text{ AND } \text{InPocket}(\text{Key})$

ב. בכדי להקל על העומס של השרטוט, נניח מספר הנחות מפשטות:
 המפתח יכול להיות רק במפסק או בכיס – ולכן $\text{InSwitch}(\text{Key}) \leftrightarrow \text{NOT}(\text{InPocket}(\text{Key}))$.
 הרכב יכול להיות רק בירושלים או בים המלח ולכן $\text{At}(\text{Car}, \text{J}) \leftrightarrow \text{NOT}(\text{At}(\text{Car}, \text{DS}))$.
 לכן, האופרטורים יראו כעת כך:

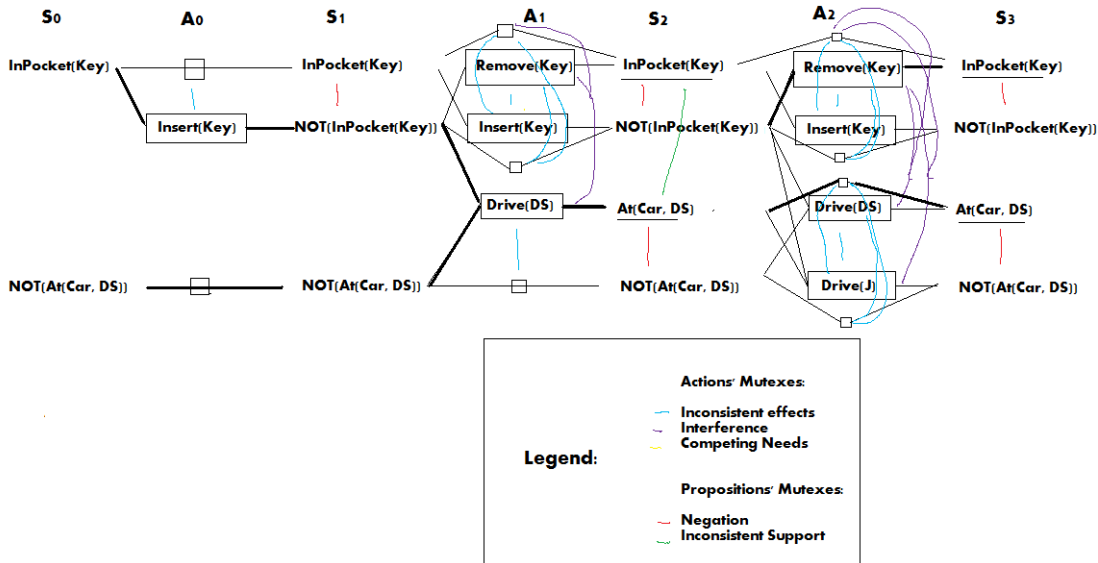
$\text{Drive}(\text{J})$: // Drive to Jerusalem
 Preconditions: $\text{At}(\text{Car}, \text{DS}) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{NOT}(\text{At}(\text{Car}, \text{DS}))$

$\text{Drive}(\text{DS})$: // Drive to the dead-sea
 Preconditions: $\text{NOT}(\text{At}(\text{Car}, \text{DS})) \text{ AND } \text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{At}(\text{Car}, \text{DS})$

$\text{Insert}(\text{Key})$: // Insert Key to switch
 Preconditions: $\text{InPocket}(\text{Key})$
 Effect: $\text{NOT}(\text{InPocket}(\text{Key}))$

$\text{Remove}(\text{Key})$: // Remove Key from switch
 Preconditions: $\text{NOT}(\text{InPocket}(\text{Key}))$
 Effect: $\text{InPocket}(\text{Key})$

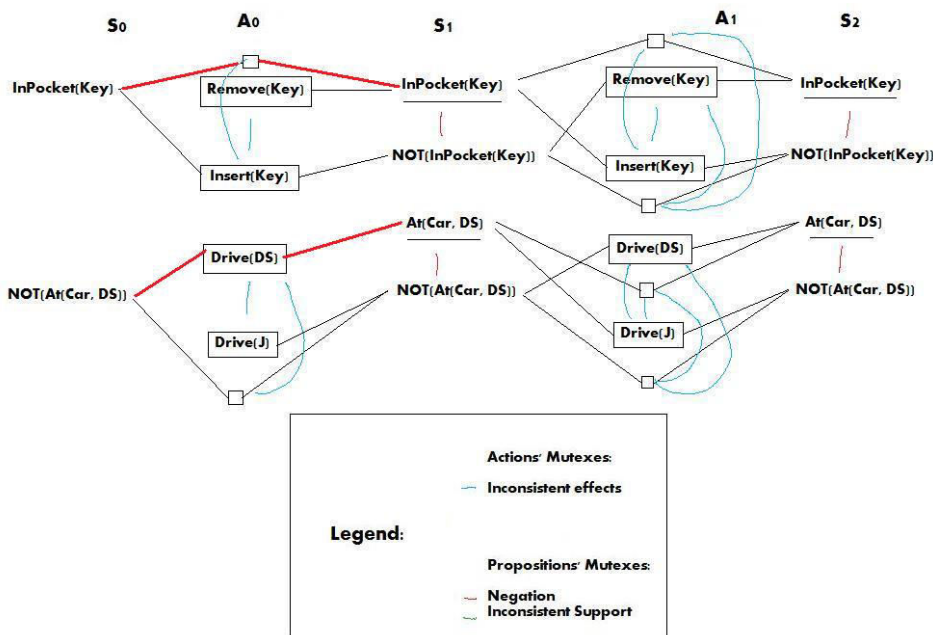
כעת נשרטט את גרף התכנון (הפתרון – מופיע בשחור מודגש):



הערה: בגרף שלהלן, Inconsistent effects mutex ו-Competing needs mutex תמיד הולכים ביחד (בגלל מבנה הבעיה הספציפי שכאן) – ולכן לא סימנתי במקרים כאלו את השני – ע"מ להקל על העומס של הגרף.
 כנ"ל ל-Negation mutex ול-Inconsistent support mutex – כשיש את הראשון – יש את השני – אם כי לא תמיד להיפך, ולכן – השמטתי את השני.
 בכל מקרה – mutex אחד בין פעולות/ליטרלים מספיק – לאלגוריתם לא משנה סוג ה-mutex, או אם קיים יותר מאחד – אלא רק האם קיים או לא.

עבור התכנית המוחלשת – נשתמש ביוריסטיקה שמתעלמת מה-pre-conditions, כאמור בעמ' 376 בספר.

גרף התכנון של הבעיה המוחלשת יראה כך:



הערה: בגרף שלהלן, אין משמעות ל-Interference mutex ול-Competing needs mutex (כי אנו מתעלמים מה-pre-conditions) – ולכן – התעלמתי מהם.
 לגבי Negation mutex ו-Inconsistent support mutex – כשיש את הראשון – יש את השני, ולכן – השמטתי את השני, ע"מ להקל על העומס של הגרף.

בכל מקרה – mutex אחד בין פעולות ליטרלים מספיק – לאלגוריתם לא משנה סוג ה-mutex, או אם קיים יותר מאחד – אלא רק האם קיים או לא. ההערכה היוריסטית (לפי היוריסטיקה הזו) של מצב ההתחלה היא 1: כשמתעלמים מה-pre-conditions, מה שנותר לעשות הוא לנסוע לים המלח מיידית – מה שניתן לעשות במקביל להכנסת המפתח ולהוצאתו מהסוויצ' (כי אנו מתעלמים מה-pre-conditions), כמופיע במסלול האדום. זה כמובן אינו האורך האמיתי של המסלול, שהנו 3 (להכניס מפתח, לנסוע לים המלח ולהוציא מפתח) – כפי שראינו בגרף המקורי (מודגש בשחור). אגב, אם לא מתעלמים מה-pre-conditions – ומחשבים לפי מס' התנאים הלא מסופקים – ומחשיבים פעולות שסותרות זו את זו (כאמור בעמ' 376 בספר) – מגיעים לערך יוריסטי של 3 – בדיוק כמו ערך המסלול האמיתי.

(1) תכנון

במשחק מגדלי הנוי עם n דסקיות בגדלים שונים. יש לסדרן על אחד העמודים בסדר יורד (כאשר הדסקית בתחתית היא הגדולה ביותר). אפשר להעביר דסקית רק אם אין מעליה דסקית אחרת. אפשר להעביר דסקית לעמוד אחר אם הוא ריק או בראשו דסקית אחרת הגדולה מהדסקית אותה רוצים להעביר. בעיה מורכבת ממצב התחלתי ומקונפיגורציה סופית של דסקיות והעמודים.

a. השלם את סט האופרטורים למטה המתאים לבעיית מגדלי הנוי פשוטה עם שתי דסקיות (BIG ו SMALL) ושלושה עמודים (משתנים מיוצגים בתוך הסוגריים המשולשים).

On(a,b) – פונק' המחזירה ערך אמת כאשר דסקית a נמצאת על עמוד b

Free(a) – פונק' המחזירה ערך אמת אין שום דסקית מעל דסקית a

Operator MOVE-SMALL <peg-x> <peg-y>

Preconditions: on (SMALL, <peg-y>), Free(Small), ~on(Big,<peg-x>)

on (SMALL <peg-x>)

not on (SMALL peg-y)

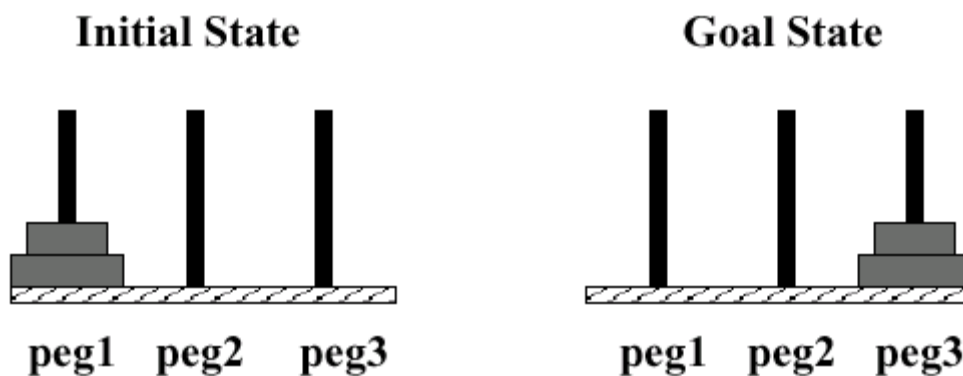
Operator MOVE-BIG <peg-x> <peg-y>

Preconditions: (BIG, <peg-y>), Free(BIG), ~on(SMALL,<peg-x>)

Add list: (BIG, <peg-x>)

Delete list: (BIG, <peg-y>)

b. רשום את התוכנית האופטימאלית לפתרון הבעיה המוצגת להלן. השתמש/י בסט אופרטורים מסעיף א'



MOVE-SMALL(peg2,peg1)

MOVE-BIG(peg3,peg1)

MOVE-SMALL(peg3,peg2)

שאלה 5 (25 נק')

בדוגמת ה- Blocks World שבספר השתמשנו בפרדיקט $On(x,y)$ לסמן שקובי ה- x נמצאת מעל עצם y

ובפרדיקט $Clear(x)$ לסמן שאין קובי ה- שיושבת מעל עצם x . העצמים שהוגדרו בעולם היו קוביות

ושולחן. הפעולה $Move(x,y,z)$ היא הזזת קובי ה- x , שנמצאת כרגע מעל קובי ה- y , להיות מעל קוביה

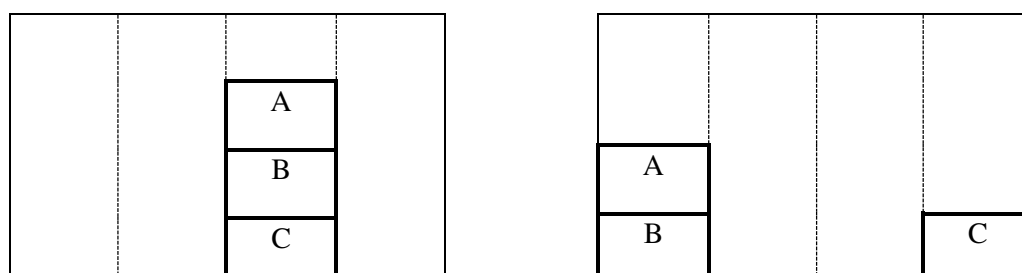
z .

א. ננתח עכשיו גרסה קצת שונה של ה- Blocks World. בגרסה זו הרצפה מחולקת ל-4 חלקים בעלי שמות שונים כאשר על כל חלק יכולות להיות מונחת

קובי ה- אחת בלבד (כמובן שמעליה יכולות להיות קוביות נוספות).

שימו לב שעכשיו העצם "שולחן" לא קיים ובמקומו יש 4 עצמים אחרים: d,e,f,g שהם חלקי השולחן.

נתאר את המצב ההתחלתי והמצב הסופי בצירור שלהלן:



תארו את המצב ההתחלתי בשפת PDDL.

On(C,f) and On(B,C) and on(A,B) and Clear(d) and Clear(e) and Clear(g)and
Clear(A)

ב. נגדיר עכשיו את האופרטור (הפעולה) Move(x,y,z) כפעולת הזזת קובי ה x, שנמצאת כרגע מעל עצם y, להיות מעל עצם z.

כתבו את סכימת הפעולה ב-PDDL.

Pre: Clear(X) and Clear(Z) and On(x,y)

Effect: Clear(Y) and Clear(X) and On(X,Z) and not (On(X,Y)) and not Clear(Z)

ג. השתמשו באלגוריתם המבצע חיפוש תוך שימוש ביוריסטיקות טובות, למציאת תכנית כדי להגיע מהמצב ההתחלתי למצב הסופי המתוארים בסעיף א'.

כתבו את התכנית הקצרה ביותר שהאלגוריתם ימצא תוך שימוש באופרטור Move(x,y,z)

בלבד.

שאלה 4: תכנון

בכיתה ראינו את ה- Blocks World כדוגמא טיפוסית לשימוש ב- STRIPS. השתמשנו בפרדיקט $On(x,y)$ לסמן שקובייה x נמצאת מעל עצם y ובפרדיקט $Clear(x)$ לסמן שאין קובייה שיושבת מעל עצם x . העצמים שהוגדרו בעולם היו קוביות ושולחן. נגדיר את האופרטור $Move(x,y,z)$ באופן הבא:

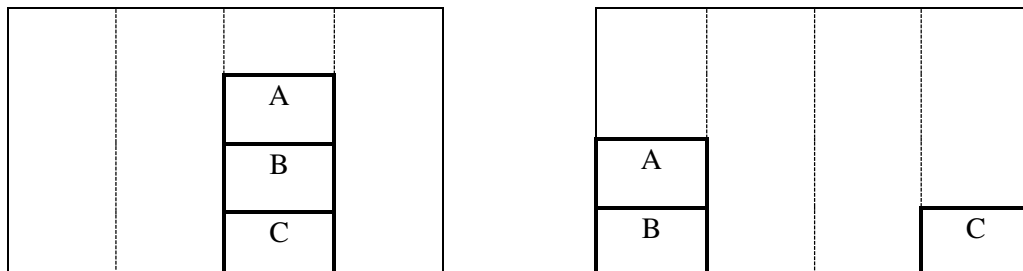
Precond [$On(x,y), Clear(x), Clear(z)$],

Add [$On(x,z), Clear(y)$],

Delete [$On(x,y), Clear(z)$]

הפעולה היא הזזת קובייה x , שנמצאת כרגע מעל קובייה y , להיות מעל קובייה z .

- (א) השלם את חלק ה- Constraints הנדרש להגדרת האופרטור.
- (ב) ננתח עכשיו גרסה קצת שונה של ה- Blocks World. בגרסה זו הרצפה מחולקת ל-4 חלקים בעלי שמות שונים כאשר על כל חלק יכולה להיות מונחת קובייה אחת בלבד (כמובן שמעליה יכולות להיות קוביות נוספות). שימו לב שעכשיו העצם "שולחן" לא קיים ובמקומו יש 4 עצמים אחרים: d, e, f, g שהם חלקי השולחן. נתאר את המצב ההתחלתי והמצב הסופי בציור:



נתאר את המצב ההתחלתי בשפת STRIPS.

- (ג) נגדיר עכשיו את אופרטור $Move(x,y,z)$ כפעולת הזזת קובייה x , שנמצאת כרגע מעל עצם y , להיות מעל עצם z . כתוב את ה- Preconditions, Add-list, Delete-list ו- Constraints הנחוצים להגדרת האופרטור, במידה והם השתנו מההגדרה הקודמת.
- (ד) נבקש מ- STRIPS למצוא תוכנית כדי להגיע מהמצב ההתחלתי למצב הסופי המתוארים בסעיף ב. כתוב את התוכנית הקצרה ביותר שהאלגוריתם ימצא תוך שימוש באופרטור $Move(x,y,z)$ בלבד (הנח שהאלגוריתם מבצע חיפוש חכם תוך שימוש בהיוריסטיקות טובות).

תשובה:

(א) $x \neq y, z$

$y \neq z$

$x, y, z \neq \text{Table}$

(ב) $\text{On}(A, B), \text{On}(B, C), \text{On}(C, f), \text{Clear}(A), \text{Clear}(d), \text{Clear}(e), \text{Clear}(g)$

(ג) ה- Preconditions, Add-list ו- Delete-list נשארו אותו דבר. רק ה- Constraints השתנו:

$x \neq y, z$

$y \neq z$

$x \neq d, e, f, g$

(ד) $\text{Move}(A, B, e)$

$\text{Move}(B, C, d)$

$\text{Move}(A, e, B)$

$\text{Move}(C, f, g)$

ניתן להחליף בסדר ביניהם



שאלה 4: Planning

נתון רובוט שפעולותיו מתוארות בטבלה הבאה:

	Go(x,y)	Pick(o)	Drop(o)
Preconditions	At(Robot,x)	At(Robot,x) ^ At(o,x)	At(Robot,x) ^ Holding(o)
Add-list	At(Robot,y)	Holding(o)	At(o,x)
Delete-list	At(Robot,x)	At(o,x)	Holding(o)

- (א) האופרטורים מאפשרים כרגע לרובוט להחזיק יותר מאובייקט אחד בו-זמנית. הראה כיצד נשנה אותם עבור רובוט שיכול להחזיק רק אובייקט אחד. היעזר בפרדיקט EmptyHand.
- (ב) האם ניתן היה לבצע את ההרחבה בסעיף הקודם ללא הפרדיקט EmptyHand? אם כן, הראה כיצד. אם לא, הסבר למה.
- (ג) נניח שהמצב ההתחלתי הוא

$At(Apple, Room1) \wedge At(Orange, Room1) \wedge At(Robot, Room1)$

ומצב המטרה הוא

$At(Apple, Room2) \wedge At(Orange, Room2)$

הראה כיצד תיראה המחסנית ומה תהיה התוכנית (החלקית) של STRIPS הרגיל ללא היוריסטיקות לאחר עשרה צעדים. השתמש בהגדרות עבור רובוט שיכול להחזיק רק אובייקט אחד שהגדרת בסעיף א'. מטרה המורכבת מ-2 מטרות או יותר תפורק תמיד כך שתת-המטרה הראשונה תהיה העליונה ביותר במחסנית, השנייה תהיה אחריה וכן הלאה. השמת אובייקט למשתנה תיעשה בזמן המאוחר ביותר (רק כשחייבים).

שאלה 4: Planning

נתון רובוט שפעולותיו מתוארות בטבלה הבאה:

	Go(x,y)	Pick(o)	Drop(o)
Preconditions	At(Robot,x)	At(Robot,x) ^ At(o,x)	At(Robot,x) ^ Holding(o)
Add-list	At(Robot,y)	Holding(o)	At(o,x)

Delete-list	At(Robot,x)	At(o,x)	Holding(o)
-------------	-------------	---------	------------

ד) האופרטורים מאפשרים כרגע לרובוט להחזיק יותר מאובייקט אחד בו-זמנית. הראה כיצד נשנה אותם עבור רובוט שיכול להחזיק רק אובייקט אחד. היעזר בפרדיקט EmptyHand.
ת: את Go אין צורך לשנות. נשנה את Pick ואת Drop

	Pick(o)	Drop(o)
Preconditions	At(Robot,x) ^ At(o,x) ^ EmptyHand	At(Robot,x) ^ Holding(o)
Add-list	Holding(o)	At(o,x) ^ EmptyHand
Delete-list	At(o,x) ^ EmptyHand	Holding(o)

ה) האם ניתן היה לבצע את ההרחבה בסעיף הקודם ללא הפרדיקט EmptyHand? אם כן, הראה כיצד. אם לא, הסבר למה.
ת: לא, מכיוון שאנו רוצים למעשה להוסיף Precondition לפעולה Pick של Holding(o) → אבל מכיוון שב- STRIPS אין שלילה של פרדיקטים אנו חייבים להשתמש בפרדיקט נוסף.
 ו) נניח שהמצב ההתחלתי הוא
 $At(Apple, Room1) \wedge At(Orange, Room1) \wedge At(Robot, Room1)$
 ומצב המטרה הוא

$$At(Apple, Room2) \wedge At(Orange, Room2)$$

הראה כיצד תיראה המחסנית ומה תהיה התוכנית (החלקית) של STRIPS הרגיל ללא היוריסטיקות לאחר עשרה צעדים. השתמש בהגדרות עבור רובוט שיכול להחזיק רק אובייקט אחד שהגדרת בסעיף א'. מטרה המורכבת מ-2 מטרות או יותר תפורק תמיד כך שתת-המטרה הראשונה תהיה העליונה ביותר במחסנית, השנייה תהיה אחריה וכן הלאה. השמת אובייקט למשתנה תיעשה בזמן המאוחר ביותר (רק כשחייבים).

ת:

#	Stack state	Actions
0	At(Apple,Room2) ^ At(Orange,Room2)	
1	At(Apple,Room2) At(Orange,Room2) At(Apple,Room2) ^ At(Orange,Room2)	
2	At(Robot,Room2) ^ Holding(Apple) Drop(Apple) At(Orange,Room2) At(Apple,Room2) ^ At(Orange,Room2)	

3	<p>At(Robot,Room2)</p> <p>Holding(Apple)</p> <p>At(Robot,Room2) ^ Holding(Apple)</p> <p>Drop(Apple)</p> <p>At(Orange,Room2)</p> <p>At(Apple,Room2) ^ At(Orange,Room2)</p>	
4	<p>At(Robot,x)</p> <p>Go(x,Room2)</p> <p>Holding(Apple)</p> <p>At(Robot,Room2) ^ Holding(Apple)</p> <p>Drop(Apple)</p> <p>At(Orange,Room2)</p> <p>At(Apple,Room2) ^ At(Orange,Room2)</p>	
5	<p>Go(Room1,Room2)</p> <p>Holding(Apple)</p> <p>At(Robot,Room2) ^ Holding(Apple)</p> <p>Drop(Apple)</p> <p>At(Orange,Room2)</p> <p>At(Apple,Room2) ^ At(Orange,Room2)</p>	
6	<p>Holding(Apple)</p> <p>At(Robot,Room2) ^ Holding(Apple)</p> <p>Drop(Apple)</p> <p>At(Orange,Room2)</p> <p>At(Apple,Room2) ^ At(Orange,Room2)</p>	Go(Room1,Room2)
7	<p>At(Robot,x) ^ At(Apple,x) ^ EmptyHand</p> <p>Pick(Apple)</p> <p>At(Robot,Room2) ^ Holding(Apple)</p> <p>Drop(Apple)</p> <p>At(Orange,Room2)</p> <p>At(Apple,Room2) ^ At(Orange,Room2)</p>	
8	<p>At(Robot,x)</p> <p>At(Apple,x)</p> <p>EmptyHand</p> <p>At(Robot,x) ^ At(Apple,x) ^ EmptyHand</p> <p>Pick(Apple)</p>	

	$\text{At}(\text{Robot}, \text{Room2}) \wedge \text{Holding}(\text{Apple})$ $\text{Drop}(\text{Apple})$ $\text{At}(\text{Orange}, \text{Room2})$ $\text{At}(\text{Apple}, \text{Room2}) \wedge \text{At}(\text{Orange}, \text{Room2})$	
9	$\text{At}(\text{Apple}, \text{Room2})$ EmptyHand $\text{At}(\text{Robot}, \text{Room2}) \wedge \text{At}(\text{Apple}, \text{Room2}) \wedge \text{EmptyHand}$ $\text{Pick}(\text{Apple})$ $\text{At}(\text{Robot}, \text{Room2}) \wedge \text{Holding}(\text{Apple})$ $\text{Drop}(\text{Apple})$ $\text{At}(\text{Orange}, \text{Room2})$ $\text{At}(\text{Apple}, \text{Room2}) \wedge \text{At}(\text{Orange}, \text{Room2})$	
10	$\text{At}(\text{Robot}, \text{Room2}) \wedge \text{Holding}(\text{Apple})$ $\text{Drop}(\text{Apple})$ EmptyHand $\text{At}(\text{Robot}, \text{Room2}) \wedge \text{At}(\text{Apple}, \text{Room2}) \wedge \text{EmptyHand}$ $\text{Pick}(\text{Apple})$ $\text{At}(\text{Robot}, \text{Room2}) \wedge \text{Holding}(\text{Apple})$ $\text{Drop}(\text{Apple})$ $\text{At}(\text{Orange}, \text{Room2})$ $\text{At}(\text{Apple}, \text{Room2}) \wedge \text{At}(\text{Orange}, \text{Room2})$	

