מודלים חישוביים - תרגיל 2#

אריאל סטולרמן

קבוצה 02

(1)

הוכחה:

תהי P בעיה סמנטית המקבלת כקלט פרדיקטים, ונוכיח כי P אינה חשיבה.

נסמן את עולם כל התוכניות שהן פרדיקטים כ-U. P מחלקת את U לשני חלקים, כך שכל שתי תוכניות p1,p2 ימצאו באותו חלק אם הם מכריעים את אותה שפה. ניתן לחלק את U למחלקות שקילות כך שכל מחלקת שקילות היא קבוצת כל הפרדיקטים המכריעים שפה מסויימת, וכל מחלקת שקילות מייצגת למעשה שפה. P מחלקת את U כך שחלק ממחלקות השקילות יהיו בצד אחד, והשאר בצד שני – חלק יקבלו T והשאר משהו אחר (F או התבדרות).

נסתכל על U לפי החלוקה של P, ונניח כי P חשיבה. ‎loopy()‎ היא סוג של פרדיקט. נניח כי P(loopy)=F, ויהי פרדיקט ‎yea()‎ הנמצא בצד השני של העולם, כלומר P(yea)=T. נכתוב כעת רדוקציה מ-$HALT_0$ ל-P:

$$HALT_0(f) := \quad P(\lambda y.\begin{cases} yea(y), & f() \\ yea(y), & o/w \end{cases})$$

הוכחת נכונות:

1. <u>התוכנית עוצרת</u>: סה"כ מפעילים את P, שהיא חשיבה לפי הנחה, על פרדיקט: התוכנית המוגדרת יכולה להתבדר או להחזיר תוצאה של yea(y) - כאשר yea פרדיקט - ולכן P אכן מופעלת על פרדיקט.
2. <u>נכונות הפלט</u>:

נניח כי f עוצרת ← ()f תעצור, ולכל y או שיוחזר yea(y) או yea(y) – כלומר התוכנית עליה מופעלת P היא למעשה yea ← P תחזיר T ← $HALT_0$ תחזיר T.

נניח כי f מתבדרת ← התוכנית עליה מופעלת P תתבדר לכל y ← התוכנית הזו שקולה ל-()loopy ולכן מכריעה את אותה שפה כמו loopy ← הפעלת P על תוכנית זה תתן את אותו פלט כמו P(loopy) שהוא F ← $HALT_0$ תחזיר F.

← סתירה, שכן $HALT_0$ אינה כריעה, ולכן גם P אינה כריעה.

הוכחה דומה ניתן לבצע למקרה בו P(loopy)=T וישנה תוכנית nay כך ש-P(nay)=F. נגדיר את $HALT_0$ כך:

$$HALT_0(f) := \quad \neg P(\lambda y.\begin{matrix} nay(y), & f() \\ nay(y), & o/w \end{matrix})$$

← כל P לפי ההגדרות בשאלה אינה כריעה ■

(2)

(a)

```
SORT = {<l> | l is a finite list of sorted numbers}
HALT-- = {<p,x> | p is a predicate program in C--, p accepts x}
```

נראה רדוקציית מיפוי $\text{HALT--} \leq_m \text{SORT}$:

נראה כי קיימת f חשיבה כך ש $f(\langle p,x\rangle) \in \text{SORT} \Leftrightarrow \langle p,x\rangle \in \text{HALT--}$:

```
f(<p,x>):=      if p(x) = T then return list(1,2,3)
                else return list(1,3,2)
```

הוכחת נכונות:

1. <u>f חשיבה ועוצרת</u>: השפה C-- היא שפה של תוכניות שתמיד עוצרות (הוכח בהרצאה), ולכן כל הפעלה של p שהיא תוכנית ב-C-- תעצור (ותחזיר T, F או תתבדר). נכונות שאר הקוד טרוויאלית.
2. <u>נכונות הפלט</u>:

<u>"←"</u>: יהי קלט $\langle p,x\rangle \in \text{HALT--}$ ← p מקבלת את x ← $p(x)=T$ ולכן f תחזיר רשימה (1,2,3) ← פלט f הוא רשימה ממויינת ← $f(\langle p,x\rangle) \in \text{SORT}$.

<u>"→"</u>: יהי קלט $\langle p,x\rangle \notin \text{HALT--}$ ← p לא מקבלת את x ← $p(x)=F \vee p(x)=\perp$ ← מוחזרת הרשימה (1,3,2) ← הפלט של f אינו רשימה ממויינת ← $f(\langle p,x\rangle) \notin \text{SORT}$.

← $f(\langle p,x\rangle) \in \text{SORT} \Leftrightarrow \langle p,x\rangle \in \text{HALT--}$ ■

כיוון ש-SORT חשיבה (לכל רשימה סופית ניתן לבדוק האם היא ממויינת), אזי לפי הרדוקציה גם HALT-- חשיבה.

(b)

רדוקציה זו אינה עובדת עבור HALT כיוון שברדוקציה לעיל אנו משתמשים בעובדה שכל תוכנית ב-C-- עוצרת לכל קלט שהוא, ולכן f, שהפעלת p נמצאת בקוד שלה, חשיבה. עצם לקיחת תוכניות ב-C-- אומרת שהתוכניות עוצרות, ולכן הרדוקציה לא רלוונטית. אך אם נקח תוכניות לא ב-C--, f לא תהיה חשיבה.

(c)

תהי A שפה כריעה, להלן רדוקצית מיפוי $A \leq_m \text{HALT}$:

נראה כי קיימת f כך ש- $\langle p,x\rangle := f(p') \in \text{HALT} \Leftrightarrow p' \in A$:

A כריעה ← קיימת f_A המחשבת אותה כך ש- $f_A(x)=T \Leftrightarrow x \in A$.

נגדיר את f כדלקמן:

$$f(p') := \begin{cases} \langle \lambda x.x, 10\rangle & f_A(p') \\ \langle \lambda x.\text{loopy}(), 10\rangle & \text{o/w} \end{cases}$$

הוכחת נכונות:

- <u>f חשיבה</u>: A שפה כריעה ולכן לכל $p' \in A$, $f_A(p')$ יקבל ערך T או F. בשני המקרים, מוחזרת תשובה חוקית וחשיבה, זוג סדיר של פונקציה כשפת תכנות כלשהי והקלט שלה (10).

- <u>נכונות הפלט</u>:

<u>"$\Rightarrow$"</u>: יהי קלט $p'\in A$ $\Leftarrow$ $f_A(p')=T$ $\Leftarrow$ $f(p')=<\lambda x.x,10>$. $\lambda x.x$ הינה תוכנית העוצרת עבור כל קלט, בפרט עבור 10 $\Leftarrow$ $f(p')=<\lambda x.x,10>\in HALT$.

<u>"$\Leftarrow$"</u>: יהי קלט $p'\notin A$ $\Leftarrow$ $f_A(p')=F$ $\Leftarrow$ $f(p')=<\lambda x.loopy(),10>$. $\lambda x.loopy()$ אינה עוצרת עבור שום קלט, בפרט עבור 10 $\Leftarrow$ $f(p')=<\lambda x.loopy(),10>\notin HALT$.

$\Leftarrow$ $p'\in A \Leftrightarrow <p,x>:=f(p')\in HALT$ ■

הערה: רדוקציה זו לא אומרת לנו כלום על HALT.

(d)

הטענה אינה נכונה:

נגדיר את השפה המשלימה של HALT:

NOT-HALT:= {<p,x> | p is a program that doesn't halt on x}

נראה רדוקציה מ-HALT אל NOT-HALT:

$HALT(<p,x>) := \neg NOT\text{-}HALT(<p,x>)$

נכונות רדוקציה זו מיידית. כעת נראה כי לא תתכן רדוקצית מיפוי מ-HALT אל NOT-HALT. נניח כי קיימת רדוקצית מיפוי. ידוע לנו כי $HALT\in RE$, ולכן $NOT\text{-}HALT\in CO\text{-}RE$. כעת, אם קיימת רדוקצית מיפוי $HALT \le_m NOT\text{-}HALT$, אז כיוון שרדוקצית מיפוי מעבירה REיות ו-CO-REיות, HALT תהיה גם CO-RE. כיוון ש-HALT היא RE וגם CO-RE אזי $HALT\in R$ – **סתירה**. לכן לא קיימת רדוקצית מיפוי במקרה זה, ולכן הטענה אינה נכונה.

(3)

$s(n):= \max\{f_i(j) \mid i,j\le n\} + 1,\ \ f_i:N\rightarrow N$

טענה: s אינה חשיבה.

הוכחה:

נניח כי s חשיבה. כיוון ש-s היא פונקציה מ-N ל-N, אזי קיים m טבעי כך ש- $s\equiv f_m$. נבחר k כלשהו כך ש- $k>m$, וקיימים p,q כך ש- $s(k)=f_m(k)=f_p(q)+1$ $\Leftarrow$ <u>$s(k)>f_p(q)$</u>.

כמו כן מתקיים: $s(k)=f_m(k)=\max\{f_i(j) \mid i,j \le k\} + 1 = f_p(q)+1$ $\Leftarrow$ לכל $i,j\le k$ מתקיים $f_i(j)\le f_p(q)$, וכיוון ש-$k>m$, בפרט מתקיים: <u>$s(k)= f_m(k)\le f_p(q)$</u>.

$\Leftarrow$ סתירה, לכן s אינה חשיבה ■

(4)

(a)

$L_1\in R,\ L_2\in R \Leftarrow L_1\cup L_2\in R$:

יהיו $L_1,\ L_2\in R$ $\Leftarrow$ קיימות $P_{L1},\ P_{L2}$ תוכניות חשיבות המכריעות את $L_1,\ L_2$. נגדיר לפיכך את התוכנית הבאה:

$P_{L1\cup L2}:= \lambda x.P_{L1}(x) \vee P_{L2}(x)$

הוכחת נכונות:

התוכנית $P_{L1\cup L2}$ חשיבה כיוון שהיא פשוט מפעילה את $P_{L1}(x)$ ו- $P_{L2}(x)$, שהן חשיבות ע"פ הנחה.

נוכיח כעת כי $P_{L1\cup L2}$ מכריעה את $L_1\cup L_2$:

- יהי $x\in L_1\cup L_2$ $\Leftarrow$ $x\in L_1$ או $x\in L_2$ $\Leftarrow$ $P_{L2}(x)=T \lor P_{L1}(x)=T$ $\Leftarrow$ $P_{L1\cup L2}(x)=T$.
- יהי $x\notin L_1\cup L_2$ $\Leftarrow$ $x\notin L_1$ וגם $x\notin L_2$ $\Leftarrow$ $P_{L2}(x)=F \land P_{L1}(x)=F$ $\Leftarrow$ $P_{L1\cup L2}(x)=F$.

$\Leftarrow$ $P_{L1\cup L2}(x)=T \Leftrightarrow x\in L_1\cup L_2$ $\Leftarrow$ $P_{L1\cup L2}$ מכריעה את $L_1\cup L_2$ $\Leftarrow$ $L_1\cup L_2\in R$ ■

(b)

$L_1\in R,\ L_2\in R$ $\Leftarrow$ $L_1\cap L_2\in R$:

יהיו $L_1,\ L_2\in R$ $\Leftarrow$ קיימות $P_{L1},\ P_{L2}$ תוכניות חשיבות המכריעות את $L_1,\ L_2$. נגדיר לפיכך את התוכנית הבאה:

$$P_{L1\cap L2} := \lambda x.P_{L1}(x) \land P_{L2}(x)$$

הוכחת נכונות:

התוכנית $P_{L1\cap L2}$ חשיבה כיוון שהיא פשוט מפעילה את $P_{L1}(x)$ ו- $P_{L2}(x)$, שהן חשיבות ע"פ הנחה.

נוכיח כעת כי $P_{L1\cap L2}$ מכריעה את $L_1\cap L_2$:

- יהי $x\in L_1\cap L_2$ $\Leftarrow$ $x\in L_1$ וגם $x\in L_2$ $\Leftarrow$ $P_{L2}(x)=T \land P_{L1}(x)=T$ $\Leftarrow$ $P_{L1\cap L2}(x)=T$.
- יהי $x\notin L_1\cap L_2$ $\Leftarrow$ $x\notin L_1$ או $x\notin L_2$ $\Leftarrow$ $P_{L2}(x)=F \lor P_{L1}(x)=F$ $\Leftarrow$ $P_{L1\cap L2}(x)=F$.

$\Leftarrow$ $P_{L1\cap L2}(x)=T \Leftrightarrow x\in L_1\cap L_2$ $\Leftarrow$ $P_{L1\cap L2}$ מכריעה את $L_1\cap L_2$ $\Leftarrow$ $L_1\cap L_2\in R$ ■

הערות:

לסעיפים c,d נשתמש בסימון הבא: תהי w מילה בשפה כלשהי L, נסמן:

$w_{(i)}$ = התו הנמצא במקום ה-i ב-w.

$<w_{(m)}\ldots w_{(n)}>$ = מילה (מחרוזת) המורכבת מהתו ה-m-י של w עד התו ה-n-י של w.

(c)

$L_1\in R,\ L_2\in R$ $\Leftarrow$ $L_{1+2}\in R$:

$L_{1+2}=\{<a_1a_2\ldots a_ma_{m+1}\ldots a_n> \mid <a_1\ldots a_m>\in L_1,\ \ <a_{m+1}\ldots a_n>\in L_2\}$, כלומר L_{1+2} היא שפת כל שרשורי המילים מ-L_1 עם המילים מ-L_2 (בסדר זה).

יהיו $L_1,\ L_2\in R$ $\Leftarrow$ קיימות $P_{L1},\ P_{L2}$ תוכניות חשיבות המכריעות את $L_1,\ L_2$. נגדיר לפיכך את התוכנית הבאה:

```
PL1+2:= λw.     for i:=0 to length(w) do
                    if PL1(<w(0)…w(i)>) AND PL2(<w(i+1)…w(length(w))>)
                        return T;
                return F;  {יחזיר שקר אך ורק אם לא החזיר אמת עד כה}
```

הוכחת נכונות:

1. <u>P_{L1+2} חשיבה</u>: כל המילים בכל שפה הן סופיות ולכן length(w) הוא מספר טבעי לכל w, ולכן לולאת ה-for תיעצר בשלב כלשהו. כמו כן אנו מפעילים את $P_{L1}(x)$ ו- $P_{L2}(x)$, שהן חשיבות ע"פ הנחה.

2. <u>נכונות הפלט</u>:

יהי $x\in L_{1+2}$ ← קיימות מילים $w_1\in L_1,\ w_2\in L_2$ כך ש- $x=<w_1w_2>$ ← קיים i כך ש-$<x_{(0)}...x_{(i)}>=w_1$, $<x_{(i+1)}...x_{(length(x))}>=w_2$ ← לולאת ה-for בהפעלת $P_{L1+2}(x)$ תחזיר T עבור אותו i ← $P_{L1+2}(x)=T$.

יהי $x\notin L_{1+2}$ ← לא קיימות מילים $w_1\in L_1,\ w_2\in L_2$ כך ש- $x=<w_1w_2>$ ← לא קיים i כך ש-$<x_{(0)}...x_{(i)}>=w_1$, $<x_{(i+1)}...x_{(length(x))}>=w_2$ ← באף איטרציה של לולאת ה-for בהפעלת $P_{L1+2}(x)$ לא יוחזר T, ולכן נצא מלולאת ה-for בסיומה ויוחזר F ← $P_{L1+2}(x)=F$.

← $P_{L1+2}(x)=T \Leftrightarrow x\in L_{1+2}$ ← P_{L1+2} מכריעה את L_{1+2} ← $L_{1+2}\in R$ ■

(d)

$L\in R$ ← $L^*\in R$ $(L^*=\cup L^i)$:

תהיה שפה $L\in R$ ← קיימת תוכנית P_L חשיבה המכריעה אותה כך ש-$x\in L \Leftrightarrow P_L(x)=T$. נגדיר את התוכנית הבאה:

```
kleene-star(w):=
     if P(w) then return T;
     for i:=0 to length(w) do
          if P(<w(0)…w(i)> and kleene-star(<w(i+1)…w(length(w))>) then
               return T;
     return F;
```

הוכחת נכונות:

1. <u>kleene-star חשיבה</u>:

- המילים בכל שפה הן סופיות ולכן length(w) הוא מספר טבעי לכל w, ולכן לולאת ה-for תיעצר בשלב כלשהו. כמו כן נובע מכך שמספר הקריאות הרקורסיביות יהיה גם כן סופי, ולכן ניעצר מתישהו.
- אנו מפעילים את P שהיא חשיבה ע"פ הנחה.

2. <u>נכונות הפלט</u>:

תהי $w\in L^*$ ← קיימים $i_1,...,i_k$ כך ש-$w=<w_{(0)}...w_{(i1)}w_{(i1+1)}...w_{(i2)}w_{(i2+1)}...w_{(ik)}>$, ו- $w_{(0)}...w_{(i1)}$, $w_{(i1+1)}...w_{(i2)}, ..., w_{(i[k-1]+1)}...w_{(ik)}$ הן מילים ב-L. לשם נוחות נסמן: $u_k:=<w_{(i[k-1])}...w_{(i[k])}>$ ← אם נהיה בשלב i כלשהו בלולאת ה-for ומתקיים כי המילה עד התו ה-i בו אנו נמצאים היא מילה ב-L, הרקורסיה תבדוק את שארית המילה לכל צירופי המילים האפשריות (וזהו צירוף סופי כיוון ששארית המילה סופית). אם $i<i_1$, אז אמנם מצאנו מילה ב-L אך לא בטוח ששארית w תהיה צירוף כלשהו של מילים ב-L, ואז נמשיך בלולאת ה-for למילה גדולה יותר ויותר, ולכל היותר נגיע למילה u_1. כשהגענו אליה מתקיים $P(u_i)=T$ – כלומר זו מילה ב-L, וגם kleene-star על שארית w יחזיר T – כיוון שקיים צירוף $u_2,u_3,...,u_k$ של מילים ב-L, אליו נגיע מתישהו ← יוחזר T ← kleene-star(w)=T.

תהי $w\notin L^*$ ← לא קיים צירוף מילים $u_1,...,u_k$ לכל k ששרשורו יהיה המילה w ← לולאת ה-for והרקורסיה בתוכה, כמתואר לעיל, לעולם לא ימצאו חלוקה ל-w למילים שכולן ב-L ← נגיע לסוף לולאת ה-for ויוחזר F ← kleene-star(w)=F.

← $kleene\text{-}star(w)=T \Leftrightarrow w\in L^*$ ← kleene-star מכריעה את L^* ← $L^*\in R$ ■

(5)

לשם נוחות נסמן את השפה ע״פ הסעיף, למשל בסעיף (a) השפה תהיה A, ונסמן את הפוני׳ המכריעה אותה כ-f_A. כמו כן p היא תוכנית. סעיפים f, h בעמוד האחרון של התרגיל.

(a)

```
A:= {<p> | ∃x such that p halts on x}
```

טענה: A∈RE/R

נראה f_A המכריעה למחצה את A. נסמן את כל הקלטים האפשריים כ-u_i המסודרים לפי סדר אלפא-ביתי, ונחשב את f_A בשיטת הזיגזג:

```
fA(p):=    i:= 0;
           while T do
                for j:= 0 to i do
                     if stepper(p,ui,j)=T or stepper(p,uj,i)=T
                     then return T
                     i:= i + 1
```

הוכחת נכונות:

1. <u>f חשיבה</u>: סה״כ מריצים שני מונים ומריצים stepper על שני קלטים חוקיים.
2. <u>נכונות הפלט</u>:

נניח כי p∈A ← קיים x כך ש-p עוצרת על x ← קיימים i מספר צעדים ו-j אינדקס כך שמתישהו התוכנית תגיע להפעלת stepper(p,x,i) ויוחזר T ← $f_A(p)=T$.

נניח כי p∉A ← התוכנית לעולם לא תתקל בקלט x ומספר צעדים i כלשהם כך שהרצת stepper תחזיר T, ולכן התוכנית תתבדר ← $f_A(p)=\perp$.

← **A∈RE**

נראה כעת רדוקצית מיפוי $A \le_m$ HALT, ע״י כך שנראה כי קיימת f כך ש-$f(<p,x>)\in A \Leftrightarrow <p,x>\in HALT$:

```
f(<p,x>):= λy. if p(x)=p(x) then return 1
```

הוכחת נכונות:

1. <u>f חשיבה</u>: סה״כ העתקנו את הקוד של p והפעלנו אותה, ונכונות שאר הקוד טרוויאלית.
2. <u>נכונות הפלט</u>:

נניח כי <p,x>∈HALT ← p עוצרת על x ← תוחזר תוכנית שלכל קלט שלה תחזיר 1, כיוון שמתקיים p(x)=p(x) (חישוב זה סופי ועוצר ומתקיים) ← קיים y (כולם למעשה) כך ש- f(<p,x>) היא תוכנית שעוצרת עליהם ← f(<p,x>)∈A.

נניח כי <p,x>∉HALT ← p לא עוצרת על x ← תוחזר תוכנית שלכל קלט מתבדרת כיוון שתחשב את ערך p(x), חישוב שאינו עוצר ← לכל קלט y התוכנית f(<p,x>) לא תעצור ← f(<p,x>)∉A.

← הרדוקציה מתקיימת ולכן כיוון ש-HALT∉R גם **A∉R**

← **A∈RE/R** ■

(b)

טענה: B∈R

בסעיף זה מוצגת שאלה ללא קלט ולה מספר תשובות סופי: או שכן או שלא, ולכן כפי שנלמד בהרצאה, למרות שטענה זו טרם הוכחה/הופרכה, היא כריעה ← **B∈R** ■

(c)

```
C:= {<p,x,y> | p halts on x and doesn't halt on y XOR
               p halts on y and doesn't halt on x}
```

טענה: C לא נמצאת באף אחד מהקבוצות R, RE/R, CO-RE/R

נניח כי C∈CO-RE ונבצע רדוקצית מיפוי HALT $\le_m$ C, ע"י כך שנראה כי קיימת f כך ש- <p,x>∈HALT ⇔ f(<p,x>)∈C:

```
f(<p,x>):= <helper, x, x+"1"> where
                    helper:=λy.if y=x then p(x) else loopy()
```

הוכחת נכונות:

1. f <u>חשיבה</u>: נכונות כל הקוד מיידית. נציין כי אם x הוא מספר אז "1"+ משמעו תוספת 1 למספר, ואם הוא מחרוזת, משמעו שרשור לתו "1" (כמו ב-Java למשל).
2. <u>נכונות הפלט</u>:

נניח כי <p,x>∈HALT ← p עוצרת על x ← helper תריץ עבור x את p(x) ותעצור, ועבור x+1 את loopy ותתבדר ← <helper, x, x+"1">∈C.

נניח כי <p,x>∉HALT ← p לא עוצרת על x ← helper תריץ עבור x את p(x) ותתבדר, ועבור x+1 את loopy וגם תתבדר ← <helper, x, x+"1">∉C.

← הרדוקציה מתקיימת ולכן כיוון ש-C∈CO-RE ע"פ הנחה, גם HALT∈CO-RE. ידוע כי HALT∈RE ← HALT∈R ← סתירה, שכן HALT אינה כריעה ← **C∉CO-RE**.

נניח כעת כי C∈RE ונבצע רדוקצית מיפוי NOT-HALT $\le_m$ C (עבור הגדרת NOT-HALT משאלה 2 סעיף d), ע"י שנראה כי קיימת f כך ש-f(<p,x>)∈C ⇔ <p,x>∈NOT-HALT:

```
f(<p,x>):= <helper, x, x+"1"> where
                    helper:=λy.if y=x then p(x) else return 1
```

הוכחת נכונות:

1. f <u>חשיבה</u>: נכונות כל הקוד מיידית, בדיוק כמו קודם.
2. <u>נכונות הפלט</u>:

נניח כי <p,x>∈NOT-HALT ← p לא עוצרת על x ← helper תריץ עבור x את p(x) ותתבדר, ועבור x+1 תחזיר 1, כלומר תעצור ← <helper, x, x+"1">∈C.

נניח כי $<p,x>\notin$NOT-HALT ← p עוצרת על x ← helper תריץ עבור x את p(x) ותעצור, ועבור x+1 תחזיר 1 וגם תעצור ← $<helper, x, x+"1">\notin C$.

← הרדוקציה מתקיימת ולכן כיוון ש-$C\in RE$ ע״פ הנחה, גם $NOT\text{-}HALT\in RE$. ידוע כי $NOT\text{-}HALT\in CO\text{-}RE$ ← $NOT\text{-}HALT\in R$ ← סתירה, שכן NOT-HALT אינה כריעה ← **$C\notin RE$.**

← $C\notin RE$ ∧ $C\notin CO\text{-}RE$ ← $C\notin R$ ← C לא שייכת לאף אחד מהקבוצות הנתונות בשאלה ■

(d)

$D:= \{<p> \mid |L(p)| > 3\}$

טענה: $D\in RE/R$

נראה f_D המכריעה למחצה את D. נסמן את כל הקלטים האפשריים כ-u_i המסודרים לפי סדר אלפא-ביתי, ונחשב את f_D בשיטת הזיגזג:

```
fD(p):=    i, count:= 0;
           while T do
                for j:= 0 to i do
                     if stepper(p,ui,j)=T then
                          if p(ui)=T then count:= count + 1
                     if stepper(p,uj,i)=T then
                          if p(uj)=T then count:= count + 1
                     if count > 3 return T
                i:= i + 1
```

הוכחת נכונות:

1. f <u>חשיבה</u>: סה״כ מריצים שלושה מונים ומריצים stepper על שני קלטים חוקיים.
2. <u>נכונות הפלט</u>:

נניח כי $p\in D$ ← קיימים לפחות 4 קלטים עבורם ערך p הוא T ← קיימים i_0, i_1, i_2, i_3 ו-j_0, j_1, j_2, j_3 עבורם הרצת stepper תעצור **וגם** $p(u_k)=T$ (מספר הצעדים ש-p תרוץ עד החזרת קלט יהיה קטן או שווה ל-j_k) ← ערכו של count יגדל ויגיע בשלב מסויים ל-4 ← יוחזר T ← $f_d(p)=T$.

נניח כי $p\notin D$ ← stepper לא יחזיר T לשום i, j טבעיים, ולכן לעולם לא יתאפשר ל-count להעלות את ערכו והתוכנית תתבדר ← $f_D(p)=\perp$.

← f_D מכריעה למחצה את D ← **$D\in RE$.**

כעת נראה כי $D\notin R$. נראה רדוקצית מיפוי $HALT \le_m D$, ע״י שנראה כי קיימת f כך ש- $<p,x>\in HALT \Leftrightarrow f(<p,x>)\in D$:

$f(<p,x>):= \lambda y.$ if p(x)=p(x) return 1

הוכחת נכונות:

1. <u>f חשיבה</u>: סה״כ העתקנו את p והפעלנו אותה על x. נכונות שארית הקוד טרוויאלית.
2. <u>נכונות הפלט</u>:

נניח כי $<p,x>\in HALT$ ← p עוצרת על x ← החישוב p(x)=p(x) עוצר ומחזיר T ולכן הפונקציה המוגדרת תמיד תחזיר 1 ← f מחזירה את הפונקציה הקבועה המחזירה 1 לכל קלט y, בפרט עבור 4 קלטים כלשהם, כלומר $|L(\lambda y.1)|>3$ ← $f(<p,x>)\in D$.

נניח כי $<p,x>\notin HALT$ ← p לא עוצרת על x ← החישוב p(x)=p(x) יתבדר ← f מחזירה את הפונקציה הקבועה המתבדרת לכל קלט y, כלומר $|L(\lambda y\ldots)| = 0 \leq 3$ (תוכנית המתבדרת לכל קלט היא למעשה פרדיקט המכריע למחצה את השפה הריקה) ← $f(<p,x>)\notin D$.

← הרדוקציה מתקיימת ולכן כיוון ש-$HALT\notin R$ גם $\mathbf{D\notin R}$.

← $\mathbf{D\in RE/R}$ ■

(e)

$D:= \{<p> \mid |L(p)| \leq 3\}$

טענה: $\mathbf{E\in CO\text{-}RE/R}$

ברור כי E היא המשלימה של D מהסעיף הקודם, וכיוון שהוכחנו ש-$D\in RE/R$ ← $\mathbf{E\in CO\text{-}RE/R}$. (ברור כי אם $E\in R$, אז גם $D\in R$ בסתירה להוכחה בסעיף הקודם).

(g)

$G:= \{<p> \mid L(p)\in RE\}$

טענה: $\mathbf{G\in R}$

מהגדרה, L(p) היא השפה שהפרדיקט p מקבל, כלומר מכריע למחצה ← כל L(p) כריעה למחצה ע״י p ← השפה G מוכרעת ע״י הפרדיקט $\lambda p.T$ ← $\mathbf{G\in R}$ ■

(6)

(a)

$L1,L2\in RE/R$.

טענה: $\mathbf{L1\cup L2\in R}$ יכול להתקיים

נגדיר את השפות הבאות:

$L1 = \{<p> \mid p \text{ is a program with no input that halts OR } |p|\in N_{even}\}$

$L2 = \{<p> \mid p \text{ is a program with no input that halts OR } |p|\in N_{odd}\}$

ברור כי L1, L2 הן ב-RE/R, אך בכל זאת בה״כ נראה כי $L1\in RE/R$: נראה תוכנית f המכריעה אותה למחצה:

```
f(p):= if |p|∈Neven OR p()=p() then return T
```

הוכחת נכונות:

1. <u>f חשיבה</u>: סה״כ העתקנו את הקוד של p והפעלנו אותה, נכונות שאר הקוד טרוויאלית.

2. <u>נכונות הפלט</u>:

נניח כי $p\in L1$ ← או שאורך התוכנית זוגי, או שההפעלה של התוכנית היא סופית ועוצרת, ואז כמובן יתקיים כי $p()=p()$ ויוחזר T ← $f(p)=T$.

נניח כי $p\notin L1$ ← אורך התוכנית אינו זוגי וגם התנאי השני לא מתקיים – בבדיקת התנאי השני נחשב את $p()$ ונתבדר ← $f(p)=\perp$.

← $L1\in RE$.

כמו כן נראה כי $L1\notin R$. נראה רדוקצית מיפוי $HALT\varepsilon \leq_m L1$, ע״י כך שנראה f המקיימת $<p>\in HALT\varepsilon \Leftrightarrow f(<p>)\in L1$:

```
f(<p>):=  if |p|∈N_even then return "λyy.p()"
          else return "λy.p()"
```

הוכחת נכונות:

1. <u>f חשיבה</u>: סה״כ בודקים אורך תוכנית סופית, מעתיקים את הקוד של p ומפעילים אותה.
2. <u>נכונות הפלט</u>:

נניח כי $p\in HALT\varepsilon$ ← p עוצרת ← אם p באורך זוגי, תוחזר תוכנית באורך אי זוגי (נוספו לה 7 תוים, בנוסף לקוד של p) שלכל y עוצרת, ואם p באורך אי זוגי, גם תוחזר תוכנית באורך אי-זוגי (נוספו לה 6 תוים) שלכל y עוצרת ← $f(<p>)$ הוא תוכנית עוצרת לכל קלט וגם באורך אי זוגי ← $f(<p>)\in L1$.

נניח כי $p\notin HALT\varepsilon$ ← p מתבדרת ← אם p באורך זוגי, תוחזר תוכנית באורך אי זוגי (נוספו לה 7 תוים, בנוסף לקוד של p) שלכל y מתבדרת, ואם p באורך אי זוגי, גם תוחזר תוכנית באורך אי-זוגי (נוספו לה 6 תוים) שלכל y מתבדרת ← $f(<p>)$ הוא תוכנית מתבדרת לכל קלט וגם באורך אי זוגי ← $f(<p>)\notin L1$.

← הרדוקציה מתקיימת, וכיוון ש-$HALT\varepsilon\notin R$, גם **$L1\notin R$**.

← **$L1\in RE/R$**. באופן דומה ניתן להוכיח ל-L2. נסתכל כעת על $L1\cup L2$:

$L1\cup L2 = \{<p> \mid |p|\in N \text{ OR}\ldots\} =$ שפת כל התוכניות

הפרדיקט $\lambda y.T$ מכריע את $L1\cup L2$ ← **$L1\cup L2\in R$** ■

(b)

טענה: לא יכול להתקיים

נניח בשלילה שהטענה יכולה להתקיים, מכאן כי קיימות: g1 המכריעה למחצה את L1, g2 המכריעה למחצה את L2, f1 המכריעה את $L1\cup L2$ ו-f2 המכריעה את $L1\cap L2$. בה״כ נוכיח כי $L1\in R$ ובכך נגיע לסתירה.

יהי x כלשהו, נגדיר את הפונקציה הבאה:

```
F(x):=    if f2(x) = T then return T
          if f1(x) = T then do
               for i:= 1 to ∞ do
                    if stepper(g1,x,i)=T return T
                    if stepper(g2,x,i)=T return F
          else return F
```

הוכחת נכונות:

1. F <u>חשיבה</u>: f2 מכריעה את החיתוך, ולכן תחזיר לכל x או T או F; f1 מכריעה את האיחוד, לכן תחזיר גם או T או F; התוכנית תכנס ללולאה אך ורק אם x שייך לאיחוד ולא לחיתוך, כלומר רק אם x שייך לאחת משתי השפות בדיוק, ולכן קיים i אליו נגיע עבורו stepper יחזיר T או ל-g1 או ל-g2, ו-F תעצר. לבסוף אם אף אחד מהתנאים הנ"ל לא מתקיים, F תיעצר ותחזיר F (false).
2. <u>נכונות הפלט</u>:

נניח כי $x \in L1$ ⬅ אם x בחיתוך $L1 \cap L2$, מיד יוחזר T. אם לא, כיוון שהוא ב-L1, הוא בהכרח באיחוד $L1 \cup L2$, ו-stepper עם g1 המכריע למחצה את L1 יחזיר T עבור i כלשהו ויוחזר T ⬅ F(x)=T.

נניח כי $x \notin L1$ ⬅ x לא בחיתוך ולכן נעבור לתנאי הבא. אם x באיחוד, הוא בהכרח ב-L2 כי הוא לא ב-L1, ולכן stepper עם g2 המכריע למחצה את L2 יחזיר T עבור i כלשהו ויוחזר F. אם הוא לא באיחוד, יוחזר F ⬅ F(x)=F.

⬅ $x \in L1$ ⇔ F(x)=T ⬅ F מכריעה את L1 ⬅ $L1 \in R$, בסתירה לנתון כי $L1 \in RE/R$ ⬅ **הטענה לא יכולה להתקיים** ■

(f)

$F:= \{<p> \mid p \text{ is a predicate such that } L(p)\in R\}$

טענה: F לא נמצאת באף אחד מהקבוצות R, RE/R, CO-RE/R

נוכיח כי הבעיה אינה כריעה ע"פ משפט Rice:

- f_F מקבלת תוכניות כקלט.
- השאלה אינה טרוויאלית, למשל: $\lambda x.T\in F$, $\lambda x.loopy()\notin F$.
- הבעיה היא בעיה סמנטית. יהיו p_1,p_2 פרדיקטים כך ש- $\forall x.p_1(x)=p_2(x)$ ← שני הפרדיקטים הללו מכריעים שפה כלשהי יחד / לא מכריעים שפה כלשהי יחד ← $F(p_1)=F(p_2)$.

← $F\notin R$.

נוכיח תחילה כי $F\notin RE$. בתרגול הרביעי ראינו הוכחה לרדוקצית מיפוי מ-HALT אל השפה המשלימה של F שהיא: $F'=\{<p>|L(p)\notin R\}$ (כלומר $HALT \le_m F'$). כיוון ש-$HALT\notin CO\text{-}RE$ אז לפי רדוקצית המיפוי גם $F'\notin CO\text{-}RE$ ← **$F\notin RE$**.

חסרה ההוכחה ש-$F\notin CO\text{-}RE$.

(h)

$H:= \{<p> \mid p \text{ is a program with no input that halts, } |p| < bb(1000)\}$

טענה: H לא נמצאת באף אחד מהקבוצות R, RE/R, CO-RE/R

נראה תחילה כי $H\notin R$:

תחילה נציין כי עבור כל i טבעי, מספר התוכניות בגודל i שעוצרות ומספר התוכניות בגודל i שלא עוצרות הוא סופי. נגדיר את P_i כאוסף כל התוכניות שאורכן i שעוצרות, וקבוצה זו היא כאמור סופית.

נניח כי $H\in R$ והתוכנית המכריעה אותה היא f_H ונכתוב את התוכנית הבאה:

```
F:= for i:=1 to ∞ do
      for each p∈Pi do:
          if fH(p)=F return i
```

תוכנית זו רצה על i החל מ-1, ובודקת את כל התוכניות שעוצרות מגודל i. אם $f_H(p)=F$, אז כיוון שהתוכנית p עוצרת (כך לקחנו אותה), הסיבה היחידה שיוחזר F היא אם גודלה גדול מ- $bb(1000)$. לפיכך, ה-i שיוחזר הוא למעשה תוצאת החישוב של $bb(1000)$, וזה כידוע אמור להיות לא חשיב ← סתירה, ולכן $H\notin R$.

הערה: גם תחת הנחה שלא ניתן לברור רק את אוספי התוכניות העוצרות שגודלן i, ניתן לעבוד גם על אוספי **כל** התוכניות שגודלן i – פשוט ניקח OR של כל $f_H(p)$ לכל תוכנית p שאורכה i, והרי לכל i (נגיד החל מאורך מסויים) קיימת תוכנית שעוצרת (טרוויאלי), אז רק ברגע שנגיע ל-$i=bb(1000)$, ה-OR יחזיר לנו F, וברור כי התנאי שהפיל אותו הוא פשוט שאורך כל תוכנית עבור אותו i אינו קטן מ- $bb(1000)$ ולא תנאי העצירה, וכיוון שאנו רצים באופן עוקב על i, בהכרח $i=bb(1000)$.

חסרה ההוכחה ש-$H\notin RE, CO\text{-}RE$.