מודלים חישוביים - תרגיל #3

אריאל סטולרמן
קבוצה 02

(1)

(*a*)

תיאור פורמלי:

$TM_{101} = \{Q, \Sigma = \{0,1\}, \Gamma = \Sigma \cup \{\bar{0}, \bar{1}, _\}, q_0, q_{accept}, q_{reject}, \delta\}$

where δ:

Q	"_"	0	1	$\bar{0}$	$\bar{1}$
q_0	$q_{00}, 1, R$	$q_{go-to-end}, \bar{0}, R$	$q_{go-to-end}, \bar{1}, R$	-	-
q_{00}	$q_{11}, 0, R$	-	-	-	-
q_{11}	$q_{accept}, 1, -$	-	-	-	-
$q_{go-to-end}$	$q_{move}, _, L$	$q_{go-to-end}, 0, R$	$q_{go-to-end}, 1, R$	-	-
q_{move}	$q_{move}, _, L$	$q_{move01}, _, R$	$q_{move11}, _, R$	$q_{move\bar{0}1}, 1, R$	$q_{move\bar{1}1}, 1, R$
q_{move11}	$q_{move12}, _, R$	-	-	-	-
q_{move12}	$q_{move13}, _, R$	-	-	-	-
q_{move13}	$q_{move}, 1, L$	-	-	-	-
q_{move01}	$q_{move02}, _, R$	-	-	-	-
q_{move02}	$q_{move03}, _, R$	-	-	-	-
q_{move03}	$q_{move}, 0, L$	-	-	-	-
$q_{move\bar{1}1}$	$q_{move\bar{1}2}, 0, R$	-	-	-	-
$q_{move\bar{1}2}$	$q_{move\bar{1}3}, 1, R$	-	-	-	-
$q_{move\bar{1}3}$	$q_{accept}, 1, -$	-	-	-	-
$q_{move\bar{0}1}$	$q_{move\bar{0}2}, 0, R$	-	-	-	-
$q_{move\bar{0}2}$	$q_{move\bar{0}3}, 1, R$	-	-	-	-
$q_{move\bar{0}3}$	$q_{accept}, 0, -$	-	-	-	-

הערה:

מקומות בטבלה עם "-" יכולים לקבל כל דבר, כיוון שלעולם לא נגיע אליהם.

תיאור מילולי:

- אם הקלט ריק, נכתוב 101 ונקבל.
- אם הקלט אינו ריק, נסמן את התו הראשון ונעביר את כל התוים שלושה מקומות ימינה. כאשר נעביר את התו הראשון ימינה שלושה מקומות, נעשה זאת תו"כ כתיבת "101" מאחוריו, ונקבל.

הסבר:
אם הקלט ריק, ישר עוברים לכתיבת המחרוזת. אם הקלט אינו ריק, מסמנים את התו הראשון בגג, ומתחילים את ההעברה של התוים ימינה (תוך השמת רווח במקומם המקורי) שלושה מקומות. עם סיום ההעברה נלך שמאלה עד שנתקל בתו הבא שיש להעביר, וכך הלאה עד שניתקל בתו עם גג. במקרה כזה נעביר אותו ימינה כך שנכתוב מאחורינו 101 אחרי כל שלוש ההעברות, וכשנגיע למיקום הסופי נכתוב את התו בלי גג (כדי לחזור לקלט המקורי), ונקבל.

(*b*)

תיאור מילולי:

- המכונה מחזיקה שני רגיסטרים: רגיסטר a המכיל את הקלט ורגיסטר b רגיסטר עזר המאותחל לאפס.
- על כל שניים שיש ב-a נשים אחד ב-b, ואם ניוותר עם 1 ב-a, לא נוסיף כלום ל-b. נוריד את a ונעלה את b בהתאם.
- כש-a יהיה 0, נעביר את כל b ל-a ונסיים.

תיאור פורמלי:

$$CM_{\lfloor \frac{n}{2} \rfloor} = \{\Lambda, R\}\ where\ \ \Lambda = \{inc(a), dec(a), inc(b), dec(b), goto\ l_3, goto\ l_4, l_1, l_2, l_3, l_4\}, \qquad R = \{a, b\}$$

$l_1: if\ a = 0\ goto\ l_3$

$dec(a)$

$l_2: if\ a = 0\ goto\ l_3$

$inc(b)$

$dec(a)$

$goto\ l_1$

$l_3: if\ b = 0\ goto\ l_4$

$inc(a)$

$dec(b)$

$goto\ l_3$

$l_4: halt$

(2)

(*a*)

$$L_2 = \{1^k |\ k = 3^n, n \in N\}, where\ 1^k = \underset{k\ times}{11 \ldots 1}$$

תיאור מילולי:

א. אם המחרוזת ריקה, נחזיר T.

ב. אם לא, נסמן את האיבר הראשון בסימון מיוחד: $\hat{1}$. נוסיף לשפה שלנו גם את הסימונים: $\bar{1}, \bar{\bar{1}}$.

ג. נלך מהאיבר הראשון קדימה עד ל-1 הלא מסומן הראשון, ונסמן אותו ואת הבא אחריו ב-$\bar{\bar{1}}$. אם לא היו מספיק 1'-ים לתהליך, **נדחה**. נחזור אחורה עד שניתקל ב-$\bar{1}$ או ב-$\hat{1}$.

o אם נתקלנו ב-$\bar{1}$, נחזור לשלב ג': נלך ממנו ימינה עד ה-1 הלא מסומן הראשון, ונסמן אותו ואת הבא אחריו ב-$\bar{\bar{1}}$.

- אם נתקלנו ב-$\hat{1}$, שוב נלך ימינה עד ה-1 הלא מסומן הראשון. אם לא מצאנו כזה, **נקבל**. אם כן מצאנו, נחזור שמאלה עד האיבר הראשון תוך שכל $\bar{\bar{1}}$ משתנה ל-$\bar{1}$. נחזור על התהליך ב-ג׳.

הסבר:

אם המחרוזת ריקה, ברור. אם לא, מה שאנו עושים הוא בדיקה האם מספר ה-1׳-ים שלנו הוא חזקה של 3 באופן הבא: בודקים עבור ה-1 הראשון האם יש לו עוד שני אחדים. אם כן, נבדוק עבור 3 האחדים הראשונים האם יש להם 6 אחדים, ע״י בדיקה לכל אחד מהם האם יש לו שניים. כך נמשיך, כאשר תנאי העצירה הם כאלה:

- אם עבור 1 כלשהו אותו בודקים גילינו שאין עבורו זוג 1׳-ים, סימן שמספר האחדים אינו חזקה של 3 ולכן נדחה.
- אם סיימנו בדיקה כלשהי עבור כל ה-1׳-ים בשלב מסויים, ושוב רצנו מהראשון לכיוון סוף המחרוזת וגילינו שאין עוד 1׳-ים לא מסומנים, אזי איחדנו שלשות ללא שארית, ולכן המספר הוא חזקה של 3, ולכן נקבל.

(*b*)

$$f(n) = n^2$$

תיאור מילולי:

א. נחזיק רגיסטר *io* שהוא רגיסטר הקלט/פלט, ועוד שלושה רגיסטרים לעזר: *a,b,c* כאשר *a* יצבור את הסכום שבסוף נעביר ל-*io*, *b* ישמש כעזר להגדלת *a* ו-*c* ישמש למעקב כמה פעמים הגדלנו את *a* ב-*n* (כך שכעבור *n* פעמים נקבל כי הערך של *a* יהיה n^2).

ב. כל עוד *io* אינו ריק, נגדיל את *a,b,c,* ב-1 – כלומר נביא את *a,b,c* להחזיק *n*, ואת *io* להיות 0.

ג. כאשר *io* ריק, נבדוק האם *c* ריק. אם כן **נסיים**, כיוון שזה אומר שמלכתחילה *io* היה ריק, ו-$0^2 = 0$.

ד. אם *c* אינו ריק, נקטין אותו ב-1 (כדי שנגדיל את *a* המכיל כבר *n* רק *(n-1) n* פעמים).

ה. נבדוק האם *c* ריק:

a. אם לא, נקטין את *c* ב-1, נרוקן את *b* ונגדיל את *io* ואת *a* עד ש-*b* יתרוקן. כש-*b* יתרוקן, אז *a* **גדל ב-*n*** וערכו של *io* הוא בדיוק *n*. כשנסיים זאת, נרוקן חזרה את *io* ונגדיל את *b* בהתאם. כשנחזור למצב בו *io* ריק, נחזור לתחילת שלב ה׳.

b. אם כן, נעביר את תוכן *a* ל-*io* **ונסיים**.

(3)

(*a*)

<u>נראה תחילה כי $L(TM') \subset RE$:</u>

תהי $L_1 \in L(TM')$, אזי לפי הגדרה קיימת תוכנית p המכריעה אותה למחצה כאשר p היא תוכנית הרצה במכונה מסוג TM' כמתואר בנתוני השאלה $\Leftarrow L_1 \in RE$ (מוכרעת למחצה ע״י p כלשהי).

<u>נראה כעת כי $RE \not\subset L(TM')$:</u>

תהי $L_2 = \{s \mid s\ is\ a\ string\ over\ \Sigma\ such\ that\ for\ each\ \alpha \in \Sigma\ \alpha\ appears\ in\ s; |\Sigma| = 9000\} \in RE$. כיוון ש-$L_2$ כריעה למחצה, קיימת p המכריעה אותה למחצה (בודקת לכל תו מהא״ב את מספר הופעותיו במחרוזת ושומרת את התוצאה במערך בגודל הא״ב, לאחר מכן עוברת על המערך עד סופו ומחזירה T אמ״מ אין אף מקום שערכו הוא 0), אך כיוון ש-L_2 היא שפה מעל א״ב המכיל 9000 תוים שונים, ו-p צריכה לבדוק האם כל תו ותו מופיע במחרוזת המקבלת כקלט, אזי לעולם לא יהיה ניתן לכתוב את p במכונה מסוג TM', כיוון ש-$|\Gamma| < 2008$. לכן $L_2 \notin L(TM')$, ולכן $RE \not\subset L(TM')$.

$L(TM') \subset RE \Leftarrow$.

(*b*)

<u>נראה תחילה ($RE \subset L(TM'')$:</u>

תהי $L_1 \in RE$. L_1 כריעה למחצה ולכן קיימת p המכריעה אותה. נשים לב כי כל TM ניתנת לכתיבה ע"י "TM כך שנאפשר תזוזות לצדדים רק בצעדים באורך 1. כיוון שכוח החישוב של TM ושל שפות תכנות הוא שווה (כפי שהוכח בהרצאה), אזי כל p שהיא תוכנית חשיבה בשפת תכנות כלשהי ניתנת להרצה גם על TM, בפרט התוכנית p המכריעה למחצה את $L_1 \Leftarrow L_1 \in L(TM) \subset L(TM'') \Leftarrow L_1 \in L(TM'') \Leftarrow RE \subset L(TM'')$.

<u>נראה כעת כי ($L(TM'') \subset RE$:</u>

תהי $L_2 \in L(TM'')$. מהגדרה L_2 כריעה למחצה ע"י תוכנית הרצה על מכונה מסוג "TM, ולכן $L_2 \in RE$.

$RE = L(TM'') \Leftarrow$.

(4)

(*a*) $L_a = \{< M, k > | \exists x. stepper(M, x, k-1) = T\}$ (כאן *stepper* הוא גרסה מתאימה למכונות טיורינג)

טענה: $L_a \in R$

מספיק לבדוק לכל הקלטים x המקיימים $|x| < k$, כיוון שלכל הקלטים שגודלם גדול או שווה ל-k אנו יודעים להכריע. עבור קלט כזה y נקח את y_{k-1} שהוא קלט המורכב מ-*k-1* תאיו הראשונים של y. נסתכל על y_{k-1}:

- אם M עוצרת עליו בפחות מ-k צעדים, סיימנו.
- אם M לא עוצרת עליו בפחות מ-k צעדים, אז היא בטוח לא תעצור על y בפחות מ-k צעדים, כיוון שאם פעולה שתתבצע על y אמורה לפעול שונה מאשר שתתבצע על y_{k-1}, זה אומר שהמכונה צריכה לדעת את השוני הזה, כלומר צריכה לבצע לפחות k פעולות רק כדי לקרוא את y, ולכן y כבר נפסל כקלט פוטנציאלי.

לכן, כל שנותר לעשות הוא לעבור על כל הקלטים האפשריים שגודלם קטן ממש מ-k, וזהו מספר סופי: $|\Sigma|^{k-1}$ ולהריץ עליהם $stepper(M, x, k-1)$. אם מצאנו אחד שמחזיר T, אזי קיים קלט עבורו M עוצרת בפחות מ-k צעדים, אחרת נחזיר F.

$\boldsymbol{L_a \in R \Leftarrow}$

(*b*) $L_b = \{< M, x > | \ M \ writes \ '1' \ on \ the \ tape \ during \ execution \ on \ x\}$

נראה תחילה כי $L_b \in RE$:

נגדיר תוכנית p המכריעה אותה למחצה באופן הבא: עבור קלט <M,x> נריץ את M על x כאשר בכל צעד נבדוק האם הראש כתב '1'. אם כן נחזיר T, אחרת נמשיך לצעד הבא. אם הגענו למצב סיום ו-M לא כתבה '1', נחזיר F.

ברור כי p מכריעה למחצה את L_b, שכן אם M כותבת '1' היא תחזיר T. אם M לא כותבת '1': אם היא סופית היא תחזיר F, ואם היא אינסופית היא תתבדר.

נראה כעת כי $L_b \notin R$ ע"י רדוקצית מיפוי $halt \leq_m L_b$:

f(<M,x>):=<M',x> where:

M' רצה כמו M, אך בכל מקום ש-M שמה 1, M' תשים $\bar{1}$ ותתייחס אליו בדיוק כמו 1 ב-M (למשל: אם מצב כלשהו מבצע פעולה כלשהי עבור קריאת 1 מהסרט ב-M, התנאי יוחלף בקריאת $\bar{1}$ ב-M'), ובמצב עצירה תכתוב 1.

אם $< M, x > \in halt$ אזי M עוצרת על x, ולכן מובטח ש-M' תעצור על x ותכתוב 1, לכן $f(< M, x >) \in L_b$. אם $< M, x > \notin halt$ אזי M לא עוצרת על x ולכן M', שלעולם לא כותבת 1, גם לא תעצור על x ולעולם לא תגיע למצב בו כותבת 1 בעצירה, ולכן $f(< M, x >) \notin L_b$.

$\Leftarrow$ רדוקצית המיפוי מתקיימת וכיוון ש-$L_b \notin R$ גם $halt \notin R$. $\boldsymbol{L_b \in RE/R}$.

(c) $L_c = \{< M > | \, M \in TM; \forall x. M(x) \; doesn't \; reach \; the \; |x| + 1000 \; cell\}$

נסתכל על השפה המשלימה: $\overline{L_c} = \{< M > | \, M \in TM; \exists x. M(x) \; reaches \; the \; |x| + 1000 \; cell\}$

נראה כי $\overline{L_c} \in RE/R$:

נראה תחילה כי קיימת תוכנית תוכנית p המקבלת את $\overline{L_c}$:

$p(M) :=$

$for \; i := 0 \; to \; \infty \; do:$

$n_{u_i} := |Q_M| \times |\Gamma_M|^{|u_i|+999} \times (|u_i| + 999)$

$C := CompHistory(M, u_i, n_{u_i} + 1)$

$for \; j := 1 \; to \; (n_{u_i} + 1) \; do:$

$if \; HeadPosition\left(\pi_j(C)\right) = |u_i| + 1000 \; then \; retun \; T$

Where:

$CompHistory(M, x, k)$ - מחזיר את ההיסטוריה החישובית של הרצת M על x עד k צעדים (יתכן פחות אם עוצרת לפני הצעד ה-k).

$HeadPosition(c)$ - מחזיר את מיקום הראש בקונפיגורציה c.

$\pi_j(C)$ - מחזיר את הקונפיגורציה ה-j בהיסטוריה החישובית C.

לכל u_i קלט אפשרי, המספר n_{u_i} הוא מספר הקונפיגורציות האפשריות בהן ראש המכונה מגיע עד התא ה-$|u_i| + 999$ (לפני שעובר אותו, כלומר לא נכללות קונפיגורציות בהן יתכן שהראש עבר תא זה, ואז חזר אליו או אף אחורה ממנו).

התוכנית p חשיבה כיוון ש-n_{u_i} הוא מספר סופי ולכן חשיב, וההיסטוריה החישובית C המחושבת גם כן סופית (תכיל $n_{u_i} + 1$ קונפיגורציות). תקינות שאר הקוד ברורה.

נניח כי קיים x עבורו M מגיעה לתא ה-$|x| + 1000$. אזי קיים i כך ש-$u_i = x$ ונריץ עליו את בדיקת ההיסטוריה החישובית של הרצת $M(u_i)$. אנו בודקים את $n_{u_i} + 1$ הקונפיגורציות הראשונות. כיוון ש-n_{u_i} הוא כל הקונ' האפשריות עד הגעת ראש המכונה לתא ה-$|u_i| + 999$, אז בספירת קונ' אחת נוספת נדע לבטח האם מגיעים לתא ה-$|u_i| + 1000$: אם לא הגענו, אז ודאי שלא הגענו לתאים רחוקים יותר, ואנו בודאי חוזרים לאחת הקונ' ב-n_{u_i} הקונ' הראשונות, וניכנס ללולאה ולעולם לא נגיע לתא ה-$|u_i| + 1000$. אם כן הגענו לתא זה, אז סיימנו ונחזיר T. אם לא, נמשיך לבדוק ונתבדר.

$\Leftarrow \overline{L_c} \in RE$. חסרה הוכחה ש-$L_c \notin R$, התשובה הסופית היא: $\boldsymbol{L_c \in co - RE/R}$.

(5)

$L \in RE \Leftarrow L \leq_m Accept$:

מיידי מרדוקצית המיפוי: נתון ש-$Accept \in RE$ ולכן גם $L \in RE$.

$L \leq_m Accept \Leftarrow L \in RE$:

נתון כי L כריעה למחצה, אזי קיימת תוכנית p המכריעה אותה למחצה. נכתוב תוכנית f כך ש-$x \in L \Leftrightarrow f(x) \in Accept$:

$$f(x) \coloneqq return < p, x >$$

f חשיבה כיוון שאנו רק מעתיקים את הקוד של p המכריעה למחצה את L, ותקינות שאר הקוד טרוויאלית.

אם $x \in L$ אזי p מקבלת את x ולכן $< p, x > \in Accept$. אם $x \notin L$ אזי p לא מקבלת את x ולכן $< p, x > \notin Accept$.

מכאן שרדוקצית המיפוי מתקיימת ולכן $L \in RE \Leftrightarrow L \leq_m Accept$.

(6)

(a)

תהיינה $L_1, L_2 \in RE$, לכן קיימות p_1, p_2 תוכניות המקבלות את L_1, L_2 בהתאם, כלומר מקיימות: $\forall x \in L_1 : p_1(x) = T$, $\forall x \in L_2 : p_2(x) = T$.

<u>סגירות RE תחת איחוד</u>:

קיימת תוכנית p כך ש-$\forall x \in L_1 \cup L_2 : p(x) = T, \forall x \notin L_1 \cup L_2 : p(x) \neq T$, המוגדרת כך:

$p(x) \coloneqq for\ i \coloneqq 0\ to\ \infty\ do$:

$$if\ stepper(p_1, x, i) = T\ OR\ stepper(p_2, x, i) = T\ then\ return\ T$$

$$i \coloneqq i + 1$$

אם $x \in L_1 \cup L_2$ אזי x נמצא לפחות באחד מבין שתי השפות, ונניח בה"כ שנמצא ב-L_1, אזי קיים i טבעי כלשהו עבורו הרצת $stepper(p_1, x, i) = T$ ולכן יוחזר T. אם $x \notin L_1 \cup L_2$ אז לא קיים i טבעי כזה, ולכן התוכנית תתבדר.

$L_1 \cup L_2 \in RE \Leftarrow$, כיוון שמוכרעת למחצה ע"י p.

<u>סגירות RE תחת חיתוך</u>:

קיימת תוכנית p כך ש-$\forall x \in L_1 \cap L_2 : p(x) = T, \forall x \notin L_1 \cap L_2 : p(x) \neq T$, המוגדרת כך:

$p(x) \coloneqq for\ i \coloneqq 0\ to\ \infty\ do$:

$$if\ stepper(p_1, x, i) = T\ AND\ stepper(p_2, x, i) = T\ then\ return\ T$$

$$i \coloneqq i + 1$$

אם $x \in L_1 \cap L_2$ אזי x נמצא בשתי השפות, ולכן קיים i טבעי כלשהו עבורו הרצת $stepper(p_1, x, i) = T$ וגם הרצת $stepper(p_2, x, i) = T$, ולכן יוחזר T. אם $x \notin L_1 \cap L_2$ אז לא קיים i טבעי כזה, ולכן התוכנית תתבדר.

$L_1 \cap L_2 \in RE \Leftarrow$, כיוון שמוכרעת למחצה ע"י p.

(*b*)

תהיינה $L_1, L_2 \in co-RE$, לכן $\overline{L_1}, \overline{L_2} \in RE$, ולכן קיימות תוכניות המקבלות את $\overline{L_1}, \overline{L_2}$ בהתאם, כלומר מקיימות:

$\forall x \in \overline{L_1}: p_1(x) = T$, $\forall x \in \overline{L_2}: p_2(x) = T$.

<u>סגירות *co-RE* תחת חיתוך:</u>

מסעיף *(a)* נובע שהשפה $L_1 \cap L_2 \in co-RE \Leftrightarrow \overline{L_1 \cap L_2} \in RE \Leftrightarrow \overline{L_1} \cup \overline{L_2} \in RE$ סגורה תחת חיתוך. ⇐ *co-RE*

<u>סגירות *co-RE* תחת איחוד:</u>

מסעיף *(a)* נובע שהשפה $L_1 \cup L_2 \in co-RE \Leftrightarrow \overline{L_1 \cup L_2} \in RE \Leftrightarrow \overline{L_1} \cap \overline{L_2} \in RE$ סגורה תחת איחוד. ⇐ *co-RE*

(*c*)

נתון כי: $A \notin R$, $A \leq_m A^c$. הטענה כי $A \notin RE, A \notin co-RE$ נכונה. הוכחה:

נניח בשלילה כי $A \in RE$, לכן $A^c \in co-RE$, אך מרדוקצית המיפוי נובע כי אז גם $A \in co-RE$. כיוון ש-$A \in RE$, אזי נקבל כי $A \in R$, בסתירה לנתון ⇐ $A \notin RE$.

נניח בשלילה כי $A \in co-RE$, לכן $A^c \in RE$, אך מרדוקצית המיפוי נובע כי אז גם $A \in RE$. כיוון ש-$A \in co-RE$, אזי נקבל כי $A \in R$, בסתירה לנתון ⇐ $A \notin co-RE$.

⇐ הטענה נכונה.

(*7*)

נתון כי $RE \ni L_1, L_2 \subseteq \{0,1\}^*$, $L_1 \cup L_2 = \{0,1\}^*$ ו-$L_1 \cap L_2 \neq \emptyset$. נוכיח כי קיימת רדוקצית מיפוי $L_1 \leq_m L_1 \cap L_2$:

- משאלה 6 סעיף *(a)* ידוע כי כיוון ש-$L_1, L_2 \in RE$ אזי גם $L_1 \cap L_2 \in RE \Leftarrow$ קיימת תוכנית p המכריעה למחצה את $L_1 \cap L_2$. נציין שוב כי p_1, p_2 מכריעות למחצה את L_1, L_2 בהתאמה.
- כמו כן ידוע כי $L_1 \cup L_2 = \{0,1\}^*$, ולכן כל מחרוזת שהיא המורכבת מ-$\{0,1\}$ תהיה באחת מהקבוצות: $L_1, L_2, L_1 \cap L_2$. לכן, אם מחרוזת כלשהי אינה ב-L_1, $L_1 \cap L_2$, אזי הוא בטוח ב-L_2.
- נציין כי ניתן לסדר את כל המחרוזות מעל $\{0,1\}$ לפי סדר לקסיקוגרפי כך ש-$u_0 = '', u_1 = '0', u_2 = '1', u_3 = '00', \dots$ וכן הלאה (ניתן לכתוב אנומרטור $g: \mathbb{N} \to \{0,1\}^*$ הממפה לקסיקורגרפית את $\{0,1\}^*$ אל הטבעיים, בשביל לחשב את כל u_i).

נגדיר את תוכנית העזר $zigzag()$:

$$zigzag() \coloneqq for\ i \coloneqq 0\ to\ \infty\ do:$$
$$for\ j \coloneqq 0\ to\ i\ do:$$
$$for\ k \coloneqq 0\ to\ i\ do:$$
$$if\ stepper(p, u_j, k) = T\ return\ u_j$$

כעת נגדיר את התוכנית f הבאה:

$f(x) \coloneqq i:=0$

$while\ T\ do$:

$if\ stepper(p_1, x, i) = T\ then\ return\ zigzag()$

$if\ stepper(p_2, x, i) = T\ then\ return\ x$

$i \coloneqq i + 1$

נניח כי $x \in L_1$, $x \notin L_1 \cap L_2$, אזי קיים i טבעי עבורו $stepper(p_1, x, i) = T$ כיוון ש-p_1 מקבלת את x, ו-$x \notin L_2$ ולכן ה-*stepper* השני לעולם לא יהיה אמת. לכן, תוחזר תוצאת הרצת *zigzag* כאשר הרצת תוכנית זו בשיטת הזיגזג **בודאות** תחזיר מחרוזת כלשהי מ-$\{0,1\}^*$ כיוון שמחפשת איבר עבורו p (המכריעה את $L_1 \cap L_2$) מקבלת, והרי נתון שהחיתוך לא ריק, לכן בטוח יוחזר איבר מהחיתוך $\Leftarrow f(x) \in L_1 \cap L_2$.

נניח כי $x \in L_1 \cap L_2$, אזי $x \in L_1, x \in L_2$, וקיים i טבעי כלשהו עבורו יתקיים אחד מה-*stepper*: אם מתקיים קודם הראשון, אז יוחזר איבר מהחיתוך, ואם מתקיים השני, יוחזר x – שהוא איבר מהחיתוך. כך או כך נקבל: $f(x) \in L_1 \cap L_2$.

נניח כי $x \notin L_1$, אזי ה-*stepper* הראשון לעולם לא יתקיים לשום i טבעי, וכיוון ש-$L_1 \cup L_2 = \{0,1\}^*$, בודאות מתקיים כי $x \in L_2$, ולכן קיים i טבעי עבורו יתקיים ה-*stepper* השני, ויוחזר x. x כאמור אינו איבר ב-$L_1, L_1 \cap L_2$, ולכן $f(x) \notin L_1 \cap L_2$.

$\Leftarrow$ רדוקציית המיפוי מתקיימת: $L_1 \leq_m L_1 \cap L_2$.