מודלים חישוביים - תרגיל 1#

אריאל סטולרמן
קבוצה 02

(1)

$$minProg(n) := \min\{|p| : p \in L, [p()] \geq n\}$$

נוכיח בשלילה כי minProg אינה חשיבה.

נניח כי minProg חשיבה, ונניח כי התוכנית M מחשבת אותה, ונסמן את אורך התוכנית $|M|=N$. כעת נגדיר את התוכנית C באופן הבא:

```
C():=
     M(n):= ... {M הגדרת}
     k:=0
     while M(k) < 5{N כתיבה מפורשת של המספר} do k:= k + 1
     return k
```

הוכחת נכונות:

1. <u>התוכנית עוצרת</u>: M מוגדרת היטב שכן ע"פ ההנחה MinProg חשיבה. כמו כן, כיוון ש-MinProg פונקציה מונוטונית עולה (יותר נכון לא יורדת), אז אם נרוץ עם k על הטבעיים החל מ-0, מתישהו נגיע למספר טבעי עבורו $M(k) \geq 5N$, ולכן התוכנית תמיד תעצור ותחזיר פלט.
2. <u>הסבר הסתירה</u>: הפלט של תוכנית C הינו פלט שע"פ הגדרת MinProg רק תוכנית באורך גדול או שווה ל-5N יכולה להחזיר, אך C הינה תוכנית באורך הקטן מ-5N:

$$\sim 30 + \log_{10} N + N < 5N$$

אורך התוכנית N המוגדרת בתוך C, ההופעה המפורשת של המספר N, שאר תווי התוכנית C

מכאן, קיבלנו תוכנית שאורכה קטן מ-5N אך מחזירה פלט שלא אמורה להיות מסוגלת להחזיר

← MinProg אינה חשיבה ■

(2)

(a) רדוקציה מ-BB ל-Halt:

ידוע כי BB אינה חשיבה, ונניח בשלילה כי Halt חשיבה. אם כן נגדיר את BB באופן הבא:

```
BB(n):=
     result:=0
     for each i:=0 to n do:
          for each w∈Li do:          {Li - i קבוצת כל המחרוזות באורך}
               if halt(w) and [w()] > result then result:= [w()]
     return result
```

הוכחת נכונות:

1. <u>התוכנית עוצרת</u>: עבור כל קלט n, מספר המחרוזות הנבדקות הוא סופי. כמו כן halt חשיבה ולכן תמיד עוצרת, והתוכנית מריצה אך ורק תוכניות שעוצרות (הערה: אם halt(w)=F נניח כי התנאי הבא לא נבדק).
2. <u>נכונות הפלט</u>: התוכנית מריצה על כל התוכניות שעוצרות ושאורכן קטן או שווה ל-n, וכל פעם מעדכנת את משתנה העזר המכיל את הפלט הגדול ביותר בו נתקלנו עד כה, אותו היא מחזירה בתום הבדיקות. כמו כן נציין כי ניתן להריץ את תוכניות ה-w העוברות את תנאי ה-if, כפי שלמדנו שקיימות תוכניות המריצות תוכניות אחרות (נראה לי ציון מיותר, אך טענה דומה נטענה בכיתה ולכן מצאתי לנכון להוסיפה). לפיכך הפלט הניתן הוא אכן הפלט הגדול ביותר הקיים עבור תוכנית כלשהי שאורכה קטן או שווה ל-n.

← מכאן ש-BB ניתנת לחישוב ע"י halt ← סתירה, שכן BB אינה חשיבה ← halt אינה חשיבה ■

מה שניתן להוכיח מרדוקציה זו הוא, כמוכח לעיל, ש-halt אינה חשיבה.

(b) רדוקציה מ-minProg ל-BB:

בכיתה תרגלנו רדוקציה מ-BB ל-minProg, להלן רדוקציה הפוכה. ידוע כי minProg אינה חשיבה, ונניח בשלילה כי BB חשיבה. אם כן נגדיר את minProg באופן הבא:

$$\texttt{minProg(n):= find(0,n) where find(k,n):=}\begin{cases}\texttt{k} & \texttt{BB(k)}\geq \texttt{n}\\ \texttt{find(k+1,n)} & \texttt{o/w}\end{cases}$$

הוכחת נכונות:

1. <u>תנאי העצירה מתקיים</u>: לכל n טבעי קיים k טבעי כלשהו כך שקיימת תוכנית כלשהי באורך k עם פלט גדול או שווה ל-n (ניתן לקחת את התוכנית הטרוויאלית המדפיסה באופן ישיר את n).
2. <u>נכונות הפלט</u>: כאשר הרקורסיה מגיעה לתנאי העצירה, מוחזר אורך התוכנית (k) עבורה הפלט המקסימלי גדול או שווה ל-n. כיוון שה-BB הינה פונקציה מונוטונית עולה, ואנו רצים על כל k טבעי החל מ-0, יתקבל כי לא קיימת תוכנית באורך פחות מ-k שהפלט שלה גדול או שווה ל-n (אחרת היינו מגיעים אליה קודם ומחזירים k קטן יותר). מכאן שאורך התוכנית המינימלי עבורו פלט התוכנית גדול או שווה ל-n, יהיה k המתקבל בתנאי העצירה.

← מכאן ש-minProg ניתנת לחישוב ע"י BB ← סתירה, שכן minProg אינה חשיבה ← BB אינה חשיבה ■

מה שניתן להוכיח מרדוקציה זו הוא, כמוכח לעיל, ש-BB אינה חשיבה.

(3)

(a) התוכנית אינה חשיבה:

נגדיר את השפה הבאה:

$$\texttt{is2008:=}\{\texttt{p: p}\in L_0,\ \texttt{[p()]== 2008}\}$$

כלומר, זוהי קבוצת כל התוכניות ללא קלט המחזירות כפלט 2008. ברור כי התוכנית הרצויה בסעיף זה היא תוכנית הבודקת שייכות לקבוצה לעיל, ומחזירה T אמ"ם התוכנית שהיא מקבלת כקלט שייכת ל-is2008, נקרא לתוכנית זו a. כעת נניח כי halt חשיבה ונגדיר רדוקצית מיפוי f מ-halt ל-is2008 אשר באמצעותה נראה כי לכל קלט $p\in L_0$ (תוכניות ללא קלט) ניתן לחשב p' כך ש- $p\in halt \Leftrightarrow p'\in is2008$.

נגדיר את f באופן הבא:

$$\texttt{f(p):=}\begin{cases}\texttt{"p'():= 2008"} & \texttt{halt(p)}\\ \texttt{"p'():= 2009"} & \texttt{o/w}\end{cases}$$

הוכחת נכונות:

1. f חשיבה: סה"כ מורכבת ממשפט תנאי פשוט, ומריצה את halt על p כלשהי, וע"פ הנחה halt חשיבה. על כן f חשיבה ותמיד תחזיר תשובה.
2. נכונות הפלט: תהי p כך ש-$p\in halt$ ← התוכנית p עוצרת ← "p'():=2008"==f(p) ← התוכנית p' שהוגדרה ע"י f מחזירה תמיד 2008 ← $p'\in is2008$.

כעת תהי p כך ש-$p\notin halt$ ← התוכנית p אינה עוצרת ← "p'():=2009"==f(p) ← התוכנית p' שהוגדרה ע"י f מחזירה תמיד 2009, כלומר לא מחזירה לעולם 2008 ← $p'\notin is2008$.

← $p\in halt \Leftrightarrow p'\in is2008$ ← מתקיימת הרדוקציה, ולכן כיוון ש-halt אינה כריעה, גם is2008 אינה כריעה, ולכן התוכנית a אינה חשיבה ■

(b) התוכנית אינה חשיבה:

נבצע רדוקציה מהתוכנית בסעיף a לתוכנית בסעיף b, אותה נכנה equals. נניח כי equals חשיבה, ונחשב את התוכנית מסעיף a באופן הבא:

```
a(p):= equals(p, const2008) where cont2008:=λx.2008
```

הוכחת נכונות:

1. התוכנית עוצרת: equals חשיבה, והתוכנית a המוגדרת לעיל פשוט מריצה אותה על שני קלטים תקינים. על כן a תמיד תעצור.
2. נכונות הפלט: תהי p תוכנית המחזירה 2008 ← ל-p ול-const2008 פלט זהה (2008) ← equals(p,const2008)=T ← a(p)=T.

תהי p תוכנית שאינה מחזירה 2008 (שעוצרת או שאינה עוצרת) ← ל-p ול-const2008 פלט שונה ← equals(p, const2008)=F ← a(p)=F.

← a ניתנת לחישוב ע"י equals ← סתירה, שכן a אינה חשיבה ← equals אינה חשיבה ■

(4)

(a)

נבצע רדוקציה מ-halt ל-MaxSteps. נניח כי MaxSteps חשיבה ונגדיר את halt באופן הבא:

```
halt(p,x):= stepper(p, x, MaxSteps(|p|))
```

Where stepper(p,x,n):=
- T running the first n steps of p(x) (or less if p.steps<n) returns an output
- F o/w

הוכחת נכונות:

1. התוכנית עוצרת:

- Stepper מוגדרת היטב ועוצרת עבור כל קלט: עבור כל תוכנית p וקלט חוקי עבור אותה תוכנית x, תמיד ניתן לרוץ מספר סופי של צעדים (n או פחות) ובסופם לבדוק האם ישנו פלט לתוכנית p(x).
- halt כמוגדר לעיל עוצרת: עבור כל קלט p, יוחזר ל-stepper מספר טבעי כלשהו, ועל כן halt תמיד תחזיר תשובה.

2. נכונות הפלט:

נניח כי התוכנית p עוצרת עבור קלט x כלשהו. לכל הפחות MaxSteps(|p|) יחזיר את מספר הצעדים בתוכנית p ← p(x) ירוץ כפי שהוא אמור במקור (כמספר הצעדים ממנו מורכבת התוכנית) ← יוחזר פלט ← stepper יחזיר T ← halt(p,x)=T.

נניח כי התוכנית p אינה עוצרת עבור קלט x כלשהו. כיוון ש-stepper מוגדר לבדוק את MaxSteps(|p|) הצעדים הראשונים של p, אנו מכסים את המקרה המקסימלי של מספר צעדים עבור כל תוכנית מהגודל של התוכנית p, ובפרט התוכנית p עצמה. לפיכך התשובה של stepper במקרה זה אמינה. כיוון שהתוכנית אינה עוצרת, היא לא תחזיר פלט עבור שום מספר צעדים שנריץ עליה ← stepper תחזיר F ← halt תחזיר F.

←מכאן ש-halt ניתנת לחישוב ע"י MaxSteps ← סתירה, שכן halt אינה חשיבה ← MaxSteps אינה חשיבה ■

(b)

נניח כי MaxSteps חשיבה. לפי סעיף (a), גם halt חשיבה. נגדיר את התוכנית הבאה למציאת פתרון למשוואה $a^n+b^n=c^n$ (עבור $n>2$):

```
FermatLastTheorem:=
     Test(a,b,n):=
          x:= n^th-root(a^n+b^n)
          if x∈N return F else Test(next(a,b,n))
     Test(1,1,3)
where next(a,b) returns the next triplet (a,b,n)∈{(x,y,i)|x,y∈N\{0}, i∈N\{0,1,2}} (by
some arrangement of that set; arrangement is possible since it's a countable set)

isFermatLastTheoremTrue?:= !halt(FermatLastTheorem)
```

התוכנית isFermatLastTheoremTrue? תחזיר T אמ"מ המשפט האחרון של פרמה הינו אמת.

הוכחת נכונות:

1. התוכנית isFermatLastTheoremTrue? עוצרת: תחת ההנחה ש-halt חשיבה (כיוון ש-MaxSteps חשיבה), כל שהתוכנית עושה הוא להריץ את halt על תוכנית כלשהי ועל כן התוכנית תמיד עוצרת.

2. נכונות הפלט:

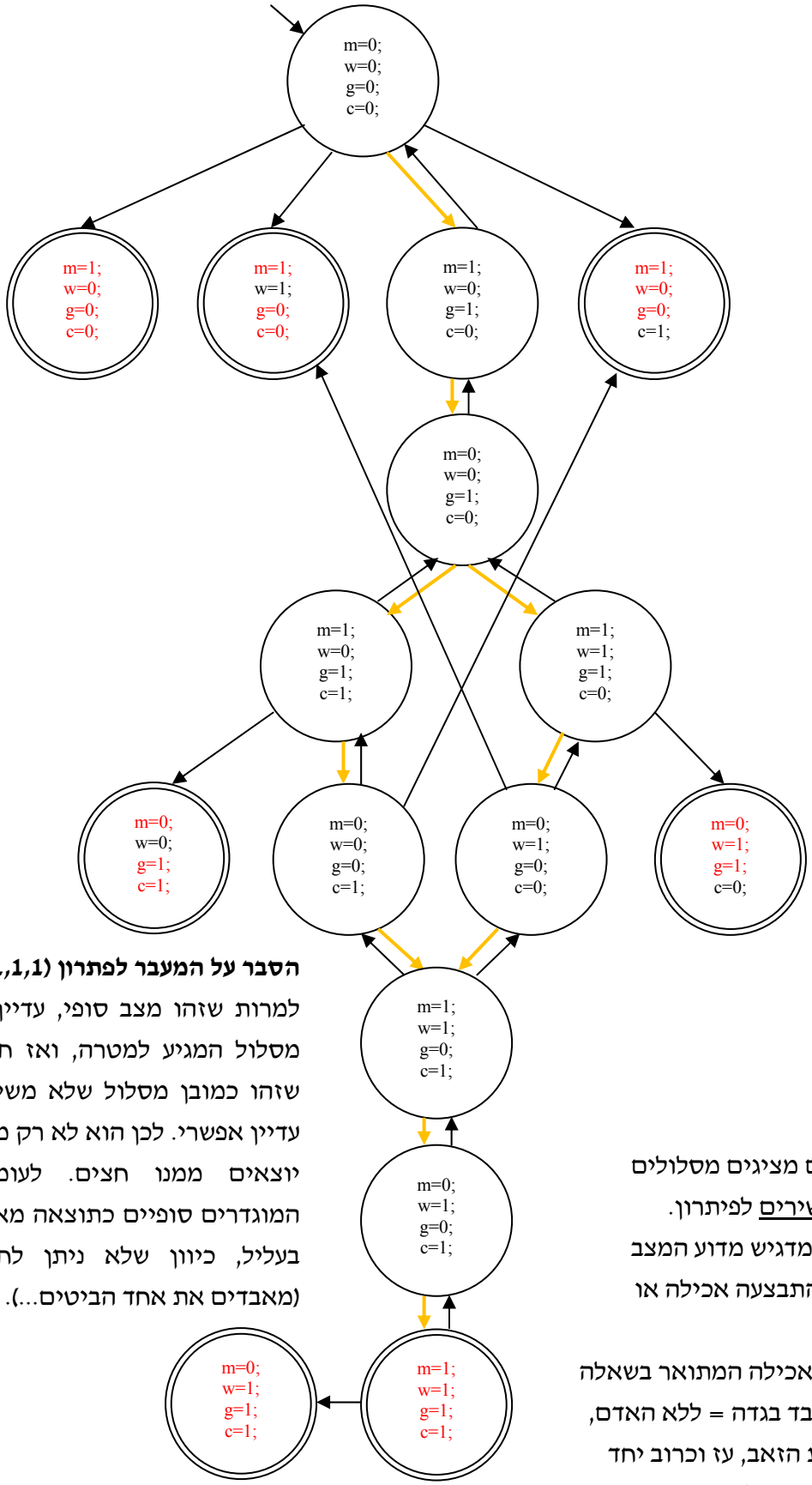
נניח כי המשפט האחרון של פרמה הינו שקר ← קיימים $a,b,c\in N\setminus\{0\}$, $n\in N\setminus\{0,1,2\}$ כך ש-$a^n+b^n=c^n$ ← תת-התוכנית test הרצה על כל השלישיות הסדורות (x,y,i) כמתואר לעיל תגיע בשלב מסויים לקלט המקיים את המשוואה ← התוכנית FermatLastTheorem תחזיר F ← התוכנית עוצרת ← isFermatLastTheoremTrue? תחזיר F.

נניח כי המשפט האחרון של פרמה הינו שקר ← הרקורסיה test בתוכנית FermatLastTheorem תרוץ לעד ← halt(FermatLastTheorm)=F ← isFermatLastTheoremTrue? תחזיר T.

← תחת ההנחה כי MaxSteps חשיבה, וניתן באמצעותה לחשב את halt, ניתן להוכיח את נכונות המשפט האחרון של פרמה ■

(5)

נסמן את המשתנים הבאים: w – לייצוג הזאב; g – לייצוג העז; c – לייצוג הכרוב; m – לייצוג האדם; כל אחד מהם יקבל 0 אם נמצא בגדה הימנית ו-1 אם נמצא בגדה השמאלית. להלן תרשים STS מתאים למשימה:



החצים הכתומים מציגים מסלולים אופציונאליים ישירים לפיתרון.
הטקסט האדום מדגיש מדוע המצב הינו מצב סופי (התבצעה אכילה או המטרה הושגה).
נתייחס לתנאי האכילה המתואר בשאלה כך שהישארות לבד בגדה = ללא האדם, ולפיכך הישארות הזאב, עז וכרוב יחד ללא האדם גוררת אכילה.

הסבר על המעבר לפתרון (1,1,1,1):

למרות שזהו מצב סופי, עדיין ניתן לקבל למשל מסלול המגיע למטרה, ואז חוזר צעד אחורה – שזהו כמובן מסלול שלא משיג את המטרה, אך עדיין אפשרי. לכן הוא לא רק מקבל חצים אלא גם יוצאים ממנו חצים. לעומת זאת, מצבים המוגדרים סופיים כתוצאה מאכילה הינם סופיים בעליל, כיוון שלא ניתן לחזור מהם אחורה (מאבדים את אחד הביטים...).

(6) להלן תיאור STS של ה-counter machine:

נתאר מצב ב-STS הרצוי כתמונת מצב של שתי שרשראות: שרשרת המקור (a) ושרשרת העזר (b), וכל מצב ייוצג ע"י זוג סדור (a,b). למשל, אם ב-a 5 חרוזים וב-b 0, הייצוג יהיה: (5,0).

$STS = \{Q, I, F, \delta\}$ כאשר:

- $I = \{(n,0)|n\in N\}$: קבוצת מצבי ההתחלה היא קבוצה אינסופית כאשר a מכילה מספר טבעי כלשהו של חרוזים ו-b מאותחלת, לכן מכילה 0 בלבד.
- $F = \{(n,0)|n\in N_{even}\}$: קבוצת מצבי הסיום היא קבוצה אינסופית כאשר a מכילה מספר טבעי כלשהו לאחר הכפלה ב-2, לכן מכילה מספר טבעי זוגי בלבד, וכיוון שסיימנו לעבוד עם b היא תכיל 0.
- $Q = \{(n,m)|n,m\in N\}$: קבוצת כל המצבים מכילה בנוסף לכל המצבים שב-I וב-Q גם את כל מצבי הביניים – בהם מורידים 1 מ-a ושמים פעמיים 1 ב-b, ונתייחס לכל פעם מהפעמיים ששמים 1 ב-b כאל מצב שונה. לפיכך, Q מכילה את כל הזוגות הסדורים מהטבעיים.
- $\delta = \{<(a,b),(x,y)>|(a,b),(x,y)\in Q,\ a=x+1 \text{ XOR } a=x-1 \text{ XOR } b=y+1 \text{ XOR } b=y-1\}$: המעברים היחידים שיכולים להיות הם המעברים הבאים:
  - הפחתנו 1 מ-a (בשלב ההכפלה – בו על כל אחד שמורידים מ-a שמים 2 ב-b).
  - הוספנו 1 ל-a (בשלב המעבר האחרון, בו מעבירים את התוצאה הסופית מ-b ל-a).
  - הפחתנו 1 מ-b (בשלב המעבר האחרון, בו מעבירים את התוצאה הסופית מ-b ל-a).
  - הוספנו 1 ל-b (בשלב ההכפלה – בו על כל אחד שמורידים מ-a שמים 2 ב-b – מעבר למצב זה מתאר את אחת ההעלאות ב-1 של b).

(7)

(a)

(i) עבור STS בעל מספר סופי של מצבים, השאלה האם יש לו ריצה מקבלת הינה שאלה **שאינה כריעה**. לכל תוכנית או אלגוריתם קיים STS המתאים להם. כעת נסתכל אך ורק על תוכניות ללא קלט, ונסתכל על ה-STS המתאימים להן.
השאלה האם ל-STS סופי כלשהו ישנה ריצה מקבלת, שקולה לשאלה האם התוכנית המתאימה ל-STS עוצרת – במקרה הכללי עבור קלט כלשהו, ובמקרה הפרטי של תוכניות ללא קלט עליו אנו מסתכלים, פשוט עוצרת. שאלה זו שקולה לפונקציה $halt_0$ עליה למדנו, הבודקת האם תוכנית ללא קלט כלשהי עוצרת או לא, ועליה ידוע והוכח **שאינה כריעה** (השפה $halt_0=\{<p>|p\in L_0,\ p \text{ halts}\}$ אינה כריעה). אם כך הוכחנו כי לפחות עבור חלק מה-STS האפשריים השאלה אינה כריעה, ולכן **השאלה אינה כריעה** ■

(ii) עבור STS בעל מספר סופי של מצבים, השאלה האם עבור n טבעי כלשהו יש לו ריצה מקבלת באורך n היא **כריעה**.
נוכיח זאת ע"י ריצה על האלגוריתם הבא:
מספר מצבי ההתחלה ב-STS סופי הינו סופי, ולכן הוא מתאים לתוכנית סופית בעלת תחום סופי של קלטים (לכל קלט מצב התחלתי שונה). עבור כל STS סופי נתון, נסמן את התוכנית המתאימה לו ב-p. בכדי לענות על השאלה הרצויה, נריץ את $stepper(p,x,n)$ לכל $x\in Dom(p)$ (כאמור תחום סופי). נוסיף להרצת stepper איזשהו counter הבודק את מספר הצעדים שביצענו, שכן יתכן ו-stepper יחזיר T באם יוחזר פלט אחרי פחות מ-n שלבים.

- עבור כל הרצה של stepper על קלט כלשהו, נבדוק האם החזיר T. אם החזיר T, נבדוק האם ה-counter שווה ל-n. אם כן, **נחזיר T** – שכן קיימת ריצה מקבלת באורך n. אם לא, נמשיך לבדיקת הקלט הבא.
- אם בסוף בדיקות כל הקלטים האפשריים (כל מצבי ההתחלה ב-STS) לא מצאנו ריצה מקבלת באורך n, **נחזיר F**.

כיוון ש-stepper היא תוכנית חשיבה, ותמיד נבדוק עבור n טבעי סופי כלשהו, האלגוריתם תמיד יסתיים. נכונות הפלט ברורה מתיאור האלגוריתם. לפיכך, **השאלה כריעה** ■

(b) **טענה: STS כמתואר לא ניתן לבנייה:**

נניח כי STS כמתואר ניתן לבנייה, אזי קיימת תוכנית כלשהי המתאימה ל-STS זה, וכיוון שקבוצת אורכי הריצות המקבלות, נסמנה L, היא בת מנייה, ניתן להניח כי תחום התוכנית עליה נסתכל מכיל קלטים היכולים להביא אותנו לריצה באורך השייך ל-L עבור כל ריצה ב-L.

אם כן, ניתן להריץ על התוכנית stepper2(p,x) על כל x בתחום של התוכנית (שסומנה p), כאשר stepper2 סופר את מספר הצעדים שעברה התוכנית p עם הקלט x עד החזרת פלט (נניח כי stepper2 עובדת על תוכניות שידוע מראש כי אינן אינסופיות, ובמקרה זה התוכנית סופית לכל קלט בתחומה שכן לכל קלט קיימת ריצה מקבלת מתאימה ב-STS).

כעת, תוצאת stepper2 תהיה k כלשהו המייצג אורך מסלול כלשהו מ-L, והוא $2^n 3^{bb(n)}$. כיוון ש-k מורכב משני מספרים ראשוניים שונים בחזקות כלשהן, ניתן לבצע עליו את הפעולות הבאות:

- נחלק ב-2 שוב ושוב עד שהמספר שנקבל לא יתחלק עוד ב-2 (יוותר רק 3 בחזקה כלשהי). מספר הפעמים שחילקנו ב-2 הוא בדיוק n.
- נחלק ב-3 שוב ושוב עד שהמספר שנקבל לא יתחלק עוד ב-3 (ניוותר עם המספר 1). מספר הפעמים שחילקנו ב-3 הוא בדיוק bb(n).

כיוון שמצאנו דרך לחשב זאת לכל n טבעי, אזי מצאנו דרך לחשב את התוכנית bb ← **סתירה**, שכן bb אינה חשיבה ← לא ניתן לבנות STS כמתואר ■