The Busy Beaver problem:  
Research Thesis in Computability

#### Advisor: Dr. Anat Paskin Cherniavsky

#### Name of Author: Or Gamliel

#### Department of Computer Sciences Ariel university

#### 12/2020

#### תוכן עניינים:

[מבוא: 3](#_Toc82956875)

[מה זה הבונה העסוק? 4](#_Toc82956876)

[ההתקדמות בבעיה עד כה 4](#_Toc82956877)

[Project poster 6](#_Toc82956878)

[Vision Statement 7](#_Toc82956879)

[How the Slowest Computer Programs Illuminate Math’s Fundamental Limits: 8](#_Toc82956880)

[סיכום מאמר "The 8000th Busy Beaver number" 11](#_Toc82956881)

["Attacking the Busy Beaver 5" 16](#_Toc82956882)

[הוכחת אי כריעות בעיית busy beaver מתוך הוכחת halting problem 20](#_Toc82956883)

[דוגמאות הרצה busy beaver 23](#_Toc82956884)

[מושגים: 25](#_Toc82956885)

[כיווני מחקר עתידיים: 27](#_Toc82956886)

[מסקנות: 27](#_Toc82956887)

[סיכום: 28](#_Toc82956888)

[ביבליוגרפיה: 29](#_Toc82956889)

# מבוא:

ה"בונה העסוק" הוא שמה של בעיה לא כריעה, מפורסמת וחשובה בחישוביות בפרט ומדעי המחשב בכלל, העוסקת בשאלה:   
"כמה פעולות ניתן לעשות באמצעות מכונת טיורינג עם מספר נתון של מצבים אפשריים, שפועלת על סרט ריק ועוצרת?".

**שתי שאלות** נגזרות מכאן:

א. מהו המספר המרבי של צעדים שמכונת טיורינג עם n מצבים יכולה לבצע? ((n)S)  
  
ב. מה הכמות המרבית של סימנים שהמכונה יכולה להדפיס לפני שהיא עוצרת? ((n)Σ)

לכן נקראת הבעיה כך, שורשה בביטוי המפורסם:  
 "busy as a beaver"

הבונים הם סמל לחריצות, כך הפונקציה החרוצה הזו שמעסיקה את מיטב המוחות בעולמנו ופתרונה עשוי לייעל הערכות/אסטימיזציות קושי לבעיות בלתי פתירות אחרות, כמו השערות גולדבך ורימן, ובאופן כללי לשפוך אור על עולם המתמטיקה, השאלות והמושגים העמוקים ביותר בו.

# מה זה הבונה העסוק?

"עוֹלָם הָפוּךְ רָאִיתִי"..  
בד"כ מתכנתים מחפשים לצמצם כמה שיותר את זמן הביצוע של הקוד שלהם, יעילות.  
אך ב1962 מתמטיקאי הונגרי בשם טיבור ראדו הציג שאלה חדשנית,  
"מה הזמן המירבי שיכולה לפעול תוכנית מחשב פשוטה לפני שתסתיים?"  
סקוט אהרונסון, מחשובי החוקרים היהודים-אמריקאים בן זמננו בנושא זה התבטא בשנינות באומרו: "במתמטיקה, יש גבול חדיר מאוד בין מה בילוי משעשע לבין מה שחשוב בפועל".

אז ראינו שהבונה העסוק עוסק בהתנהגותן של מכונות טיורינג, המחשב הפרימיטיבי אך שקול לכל מחשב מודרני המוכר לנו כיום.

אלן טיורינג שסוג פשוט של מחשב זה מסוגל לבצע כל חישוב אפשרי, בהתחשב בהוראות הנכונות ובזמן מספיק.

מכונת טיורינג מבצעת פעולות על רצועת קלטת אינסופית המחולקת לריבועים.   
היא עושה זאת על פי רשימת כללים, אך בסופו של דבר יש את ההוראה לעצור.

כדי לחשב, מכונת טיורינג חייבת בסופו של דבר להפסיק - היא לא יכולה להילכד בלולאה אינסופית. הקטע הוא שטיורינג הוכיח דבר נוסף משמעותי מאוד,  
אין שיטה אמינה וחוזרת על עצמה להבחין בין מכונות שעוצרות ממכונות שפשוט פועלות לנצח - עובדה זו מכונה בעיית העצירה.

לכן, ככל הידוע לנו זה לא ייתכן, אבל אם בעיית הבונה העסוק תיפתר, תיפתר גם בעיית העצירה, מעמודי התווך של מדעי המחשב התיאורטיים.

# ההתקדמות בבעיה עד כה

נתחיל במקרה הבסיסי ביותר, בו מספר המצבים, n=1.

כפי שראינו, נרצה להבטיח שמכונת טיורינג תיפסק, אז ניאלץ לכלול את הוראת העצירה מיד.  
מספר הבונה העמוס של מכונת בעלת כלל אחד, או יותר פורמלי  
BB (1), הוא למעשה 1.

נחדד שוב, כיוון שהוכחנו שאין שום דרך לדעת באופן אוטומטי אם מכונה שפועלת באלף או מיליון צעדים לא תסתיים בסופו של דבר (halting problem), קשה כל כך למצוא "בונים עסוקים" וצריך לבדוק כל מקרה לגופו.

קשה עד כדי כך שגם בBB(2) = 6 &BB(3) = 21 המצומצמים ביותר האפשריים, זיכו את Shen Lin, תלמידו המובהק של ראדו בדוקטורט.

נתבונן בשאר המקרים עד מכונה בעלת שישה מצבים, מה ה((n)S) וה((n)Σ בכל אחד מהם:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

# Project poster

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Vision Statement   
**Vision Statement – The Busy Beaver problem:   
arranged by: Or Gamliel**

* **Project goals** –  my project goals are:

       understanding the busy beaver problem

      trying to proceed on one of the open issues on this subject or thinking of new   
      problem by ourselves

Quality and **rare** information in **Hebrew** about computability in general and                                                                                           undecidable problems in particular.

* **Project scope** – my project will include an important data about Computability theory, which is one of the most central branches of mathematical logic and computer science.   
  The project will help to understand main problem and will offer an optional solutions.

* ***essential* and *desirable***–   
  exploring and explaining the busy beaver problem are essential.    
     
  Finding reasonable solutions and expansion in the matter of computational power, are desirable.

* **Major milestones will be:**   
     
  Understanding the problem,    
  writing it and describing it correctly,    
  thinking about other possible problems derived from it,    
  offering some optional solutions.

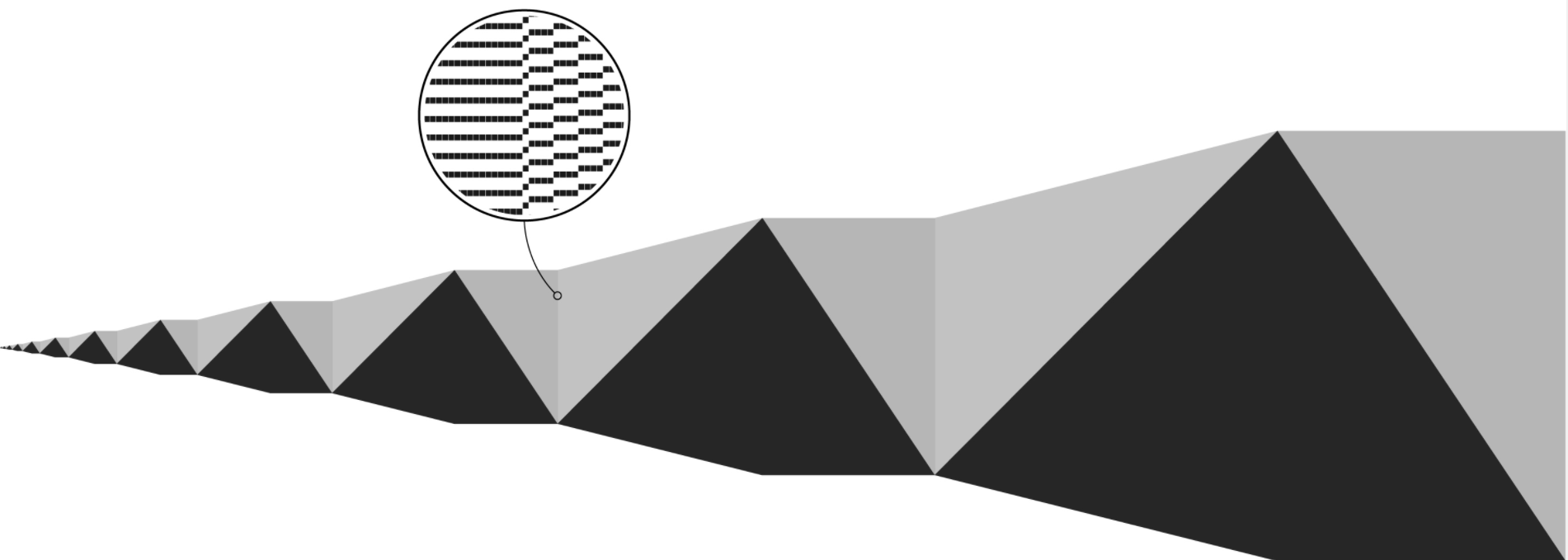
In addition, organizing, arranging and properly translating all the articles will be a decent task

* **The project** is basically for researchers and students in this field, and of course anyone who is interested in this fascinating topic and looking for information about it in Hebrew.    
  The busy beaver project is a problem in theoretical computer science, which is still subject in constant study,  so that project will be a good assistance for the people we have mentioned  before.   
  unlike other similar projects which are full of Information that is not necessarily related to the topic and not accessible to Israelis who want to read an academic material in Hebrew, I will do my best to assure that   
  my project will be well constructed, understandable to the average reader yet also in-high level in its field.

# How the Slowest Computer Programs Illuminate Math’s Fundamental Limits:

כפי שנוכחנו לדעת, מטרת המשחק "בונה עסוק" היא למצוא את תוכנית המחשב הארוכה ביותר. העיסוק שלה כולל קשרים מפתיעים לכמה מהשאלות והמושגים העמוקים ביותר במתמטיקה.

הדמיה של מכונת הטיורינג בעלת חמשת המצבים שרצה הכי הרבה, ידועה כיום.  
כל טור פיקסלים מייצג שלב אחד בחישוב, הנע משמאל לימין. ריבועים שחורים מראים היכן המכונה הדפיסה 1. העמודה הימנית ביותר מציגה את מצב החישוב כאשר מכונת הטיורינג עוצרת.



אנו יודעים כי מתכנתים בדרך כלל רוצים לצמצם למינימום האפשרי את זמן הריצה של הקוד שלהם. אך בשנת 1962 הציב המתמטיקאי ההונגרי טיבור ראדו את הבעיה ההפוכה.  
הוא שאל: כמה זמן תוכנת מחשב פשוטה יכולה לרוץ לפני שהיא מסתיימת?  
ראדו כינה את התוכניות הבלתי יעילות אך הפונקציונליות הללו בשם: "בונים עסוקים".

מציאת תוכניות אלה הייתה חידה אדישה להפליא בעבור מתכנתים וחובבים מתמטיקה אחרים עד שזכתה לפופולריות בטור "Computer Recreations" של Scientific American’s בשנת 1984.  
בשנים האחרונות הבונה העסוק למושא לימוד בפני עצמו, כיוון שהוא הניב קשרים לכמה מהמושגים הנעלים ביותר ולבעיות הפתוחות במתמטיקה.

העבודות האחרונות בנושא, מצביעות על כך שחיפוש אחר תוכנות מחשב ארוכות טווח יכול להאיר את מצב הידע המתמטי ואף לספר לנו מה או עד כמה ניתן לדעת.   
לדברי החוקרים, משחק הבונים העמוס מספק אמות מידה קונקרטיות להערכת הקושי בבעיות מסוימות, כגון השערת גולדבך הבלתי פתורה והשערת רימן. הוא אפילו מציע הצצה לאן מתפרק הסלע ההגיוני העומד בבסיס המתמטיקה.  
הלוגיקן Kurt Gödel הוכיח את קיומה של ה'terra incognita' (שדה מידע שלא נחקר) המתמטית הזו לפני כמעט מאה שנה, אבל בעיית הבונה העסוק יכולה להראות היכן הוא בעצם מונח על ציר מספרים, כמו מפה עתיקה המתארת את קצה העולם.

משחק הבונה העסוק עוסק בעיקר בהתנהגותן של מכונות טיורינג - המחשבים הפרימיטיביים והאידיאליים לייצוג הבעיה שהגה Alan Turing בשנת 1936. מכונת טיורינג מבצעת פעולות על רצועה אינסופית של קלטת המחולקת לריבועים. היא עושה זאת על פי רשימת כללים.  
הכלל הראשון עשוי לומר משהו בסגנון:  
אם הריבוע מכיל 0, החלף אותו ב- 1, זוז ריבוע אחד ימינה ובדוק את כלל 2.  
אם הריבוע מכיל 1, השאר את ה -1, זוז ריבוע אחד שמאלה ובדוק את כלל 3.

לכל כלל יש את הסגנון ההסתעפות הזה של choose-your-own-adventure. חלק מהכללים אומרים לקפוץ חזרה לחוקים הקודמים, אך בסופו של דבר חייבים שיהיה לנו כלל המכיל את ההוראה "עצור". טיורינג הוכיח שמחשב פשוט מסוג זה מסוגל לבצע כל חישוב אפשרי, בהתחשב בהנחיות הנכונות ובזמן מספיק לביצוע.

כפי שציין טיורינג בשנת 1936, על מנת לחשב משהו, מכונת טיורינג חייבת בסופו של דבר לעצור, היא לא יכולה להילכד בלולאה אינסופית. אך הוא גם הוכיח כי **אין** שיטה אמינה, הניתנת לשחזור, להבדיל בין מכונות שעוצרות ממכונות שפשוט פועלות לנצח - עובדה הידועה בשם בעיית העצירה.

לכן בא משחק הבונה העסוק ושואל: בהתחשב במספר כללים כללי כלשהוא,  
מהו מספר הצעדים המרבי שמכונת טיורינג יכולה לבצע לפני עצירה?

לדוגמה, אם מותר לנו כלל אחד בלבד ונרצה להבטיח שמכונת הטיורינג ניאלץ לכלול את הוראת העצירה מיד. תוצאת הפעלת פונ' הבונה העסוק על מכונה בעלת כלל אחד, או (1)BB, הוא אֵפוֹא 1.

אך הוספת מספר כללים בודדים נוספים מפוצצת באופן מיידי את מספר המכונות שיש לקחת בחשבון.   
מתוך 6,561 מכונות אפשריות עם שני כללים, זו שפועלת הכי הרבה זמן - שישה צעדים, לפני העצירה, היא זו של הבונה העסוק. אבל יש כמה אחרים שפשוט רצים לנצח.  
אף אחד מאלה אינו הבונה העסוק. אך כיצד ניתן לשלול אותם סופית?  
טיורינג הוכיח כי אין דרך לדעת באופן אוטומטי אם מכונה שפועלת באלף או מיליון צעדים לא תיגמר בסופו של דבר.

לכן קשה כל כך למצוא את המספר המדויק של הבונה העסוק. אין גישה כללית לזיהוי מכונות טיורינג שרצות הכי הרבה ע"י מספר הוראות שרירותי. אנו נצטרך להבין פרטי כל מקרה לגופו.  
במילים אחרות, משחק הבונה העסוק הוא, באופן כללי, "בלתי ניתן לחישוב".

ההוכחה ש- BB (2) = 6 ו- BB (3) = 21 הייתה קשה דיה בכדי שתלמידו של רדו שן לין יקבל דוקטורט על עבודתו בשנת 1965. ראדו ראה את BB(4) כ"entirely hopeless", אך בסופו של דבר ה'תיק' נפתר כמעט 20 שנה לאחר מכן. מעבר לכך, הערכים מתפוצצים ככל שעולים בכמות החוקים. חוקרים זיהו מכונת טיורינג בת חמישה כללים, שפועלת לאורך 47,176,870 צעדים לפני שעוצרת, כך ש- BB(5) היא לפחות ככה גדולה. BB(6) הוא לפחות 7.4 × 1036,534..  
הוכחת הערכים המדויקים "תזדקק לרעיונות חדשים ותובנות חדשות, אם ניתן בכלל לעשות זאת", אמר ד"ר סקוט אהרונסון, חוקר מפורסם בנושא שנסקור מאמר שלו בהמשך.

William Gasarch, מדען מחשבים מאונ' מרילנד, אמר כי הוא פחות מסוקרן מהסיכוי להצמיד למטה (חסם תחתון) מספרים של הבונה העסוק, ואילו הוא ומתמטיקאים אחרים מעוניינים בעיקר להשתמש במשחק כמדידת הקושי בבעיות פתוחות חשובות במתמטיקה - או כדי להבין מה אפשר לדעת בכלל מבחינה מתמטית.

השערת גולדבך, למשל, שואלת האם כל מספר שלם גדול מ-2 הוא סכום של שני ראשונים.  
הוכחת ההשערה כנכונה או לא נכונה, תהיה אירוע תקופתי משמעותי בתורת המספרים, המאפשר למתמטיקאים להבין טוב יותר את התפלגות המספרים הראשוניים.   
בשנת 2015, משתמש אלמוני פירסם בGitHub קוד למכונת טיורינג בעלת 27 כללים שעוצרת אם - ורק אם - השערת גולדבך היא שקר. זה פועל על ידי ספירה כלפי מעלה בכל המספרים השלמים הגדולים מ -4.  
עבור כל אחד, הוא 'טוחן' את כל הדרכים האפשריות להשיג את המספר השלם על ידי סכימת שניים אחרים, ובודק אם הצמד הוא ראשוני. כשהוא מוצא זוג ראשוני מתאים, הוא עובר למספר השלם הבא וחוזר על התהליך. אם הוא מוצא מספר שלם ש**לא** ניתן לסכום אותו ע"י צמד של מספרים ראשוניים, הוא עוצר.  
  
הפעלת המכונה חסרת הבינה הזו אינה דרך מעשית לפתור את ההשערה, מכיוון שאיננו יכולים לדעת אם היא תיעצר (halting problem) עד שתעשה זאת.  
אך משחק הבונה העסוק שופך מעט אור על הבעיה. אם היה אפשר לחשב BB(27), זה היה מספק תקרת גג לכמה זמן נצטרך לחכות עד שההשערה של גולדבך תְיֻשָּב אוטומטית.  
הסיבה לכך היא ש- (27)BB תואם את מספר הצעדים המרבי שמכונת טיורינג הכוללת 27 כללים תצטרך לבצע על מנת לעצור אותו (אם בכלל הייתה עוצרת). אם היינו יודעים את המספר הזה, היינו מפעילים את מכונת הטיורינג בדיוק במס' שלבים זה. אם זה היה נעצר בנקודה כלשהיא באמצע, היינו מבינים שהשערת גולדבך שקרית. אבל אם זה היה עובר כל כך הרבה צעדים ולא היה עוצר, היינו יודעים בוודאות שזה לעולם לא יקרה - וכך מוכיחים שההשערה נכונה.

הקושי הוא ש- BB(27) הוא מספר כה אדיר מימדים עד כדי כך שאפילו כתיבתו, ועַל אַחַת כַּמָּה וְכַמָּה הפעלת מכונת טיורינג שתבדוק את הפרכת גולדבך בשלבים כה רבים, לא תעלה על הדעת ביקום הפיזי שלנו. אף על פי כן, המספר העצום הבלתי נתפס הזה הוא עדיין נתון מדויק שעוצמתו, על פי אהרונסון, מייצגת "אמירה על הידע הנוכחי שלנו" בתורת המספרים.

בשנת 2016, אהרונסון השיג תוצאה דומה בשיתוף פעולה עם שני חוקרים נוספים.   
הם זיהו מכונת טיורינג הכוללת 744 חוקים שעוצרת אם ורק אם השערת רימן אינה נכונה.  
השערת רימן נוגעת גם היא לחלוקת המספרים הראשוניים והיא אחת משבע "בעיות המילניום" של מכון קליי בשווי מיליון דולר. המכונה של אהרונסון תספק פתרון אוטומטי בBB(744) צעדים.  
(היא פועלת על פי אותו תהליך חסר בינה כמו מכונת גולדבך, וחוזרת באיטרציות כלפי מעלה עד שהיא מוצאת דוגמה נגדית).

כמובן, BB(744) הוא מספר גדול עוד יותר ובלתי מושג מאשר (27)BB. אבל העבודה על מנת לאתר במדויק משהו קל יותר, כמו BB(5), "עשויה להעלות כמה שאלות חדשות ומעניינות בפני עצמן בתורת המספרים", אמר אהרונסון. לדוגמה, המתמטיקאי פסקל מישל הוכיח בשנת 1993 כי מכונת הטיורינג בעלת חמשת הכללים מפגינה התנהגות דומה לזו של הפונקציה המתוארת בהשערה של קולץ (השערת 3n+1), עוד בעיה פתוחה מפורסמת בתורת המספרים.  
  
"כל כך הרבה במתמטיקה יכול להיות מקודד כשאלה: 'האם מכונת הטיורינג הזו נעצרת או לא?' " אומר ד"ר אהרונסון. "אם היינו יודעים את כל המספרים של הבונה העסוק, היינו יכולים ליישב את כל השאלות הללו".  
  
לאחרונה השתמש אהרונסון בקנה מידה עסוק הנגזר על ידי הבונה העסוק בכדי לאמוד את מה שהוא מכנה "סף חוסר הידיעה" עבור מערכות מתמטיות שלמות.  
משפטי חוסר השלמות המפורסמים של גֶדֶל משנת 1931 הוכיחו ש**כל** מערכת אקסיומות בסיסיות שיכולות לשמש בסיס לוגי אפשרי למתמטיקה נידונה לאחד משני גורלות: או שהאקסיומות לא יהיו עקביות, מה שיוביל לסתירות (כמו להוכיח ש 0 = 1), או שהם לא יהיו שלמים, כלומר לא יוכלו להוכיח כמה אמירות אמיתיות לגבי מספרים (כמו העובדה ש- 4=2+2).  
למערכת האקסיומטית העומדת בבסיס כמעט כל המתמטיקה המודרנית, הידועה בשם תורת המערכות ZF, יש גבולות גֶדֶליים משלה, ואהרונסון השתמש במשחק הבונים העסוק בכדי לקבוע היכן הם נמצאים.  
  
בשנת 2016, במאמר הבא שנבחן, סקוט אהרונסון ותלמידו אדם ידידיה ציינו מכונת טיורינג בת 7,910 חוקים שתעצור רק אם תורת הקבוצות של ZF אינה עקבית. המשמעות היא ש- (7910)BB הוא חישוב המתחמק מאקסיומות של תורת הקבוצות ZF. **לא ניתן** להשתמש באקסיומות אלה כדי להוכיח כי (7910)BB מייצג מספר אחד במקום מספר אחר,   
זה שקול לחוסר מסוגלות להוכיח כי 4=2+2 ולא ל-5 לדוגמא.  
  
חוקר בשם O'Rear תכנן לאחר מכן מכונה פשוטה יותר בעלת 748 כללים שעוצרת אם ZF אינו עקבי, ובכך קירב את סף חוסר הידיעה, מ- (7,910)BB ל- (748)BB.   
"זה סוג של דרמטי, שמספר החוקים אינו מגוחך לחלוטין", אמר פרופ' הארווי פרידמן, לוגיקן מתמטי. פרידמן חושב שאפשר להוריד את המספר עוד יותר: "אני חושב שאולי 50 היא התשובה הנכונה". אהרונסון חושד כי הסף האמיתי עשוי להיות קרוב יותר, בכיוון של (20)BB.  
  
בין אם סף חוסר הידיעה קרוב או רחוק, הוא בהחלט קיים.   
"זהו חזון העולם המתמטי שיש לנו מאז גֶדֶל", אמר אהרונסון.  
"פונקציית הבונה העסוק היא דרך נוספת להפוך אותו לקונקרטי".

# סיכום מאמר "The 8000th Busy Beaver number"

פְּרֶה מאמר:  
בפוסטים מוקדמים תיאר ד"ר סקוט את מדעי המחשב התיאורטיים כ"אפיסטמולוגיה (תורת ההכרה, גבולות הידיעה) כמותית".  
בניית מכונות טיורינג קטנות שהתנהגותן חומקת מתורת הקבוצות איננה מדעי המחשב התיאורטיים המקובלים בשום דמיון.   
זה קרוב יותר בפועל לשפות תכנות או לארכיטקטורת מחשבים, או אפילו לתרגול פנאי המכונה code-golfing. כאן נבחן פרויקט אחר שעסק בצורה מאוד ברורה ומפורשת בהצבת הגבול הכמותי בין הנודע אל הלא נודע.

מאמרם של סקוט ארונסון ואדם ידידיה עוסק ב"התחמקותו" של המספר ה-8000 בבעיית הבונה העסוק מ[תאוריית ZF של תורת הקבוצות](#zermelo_frankel_set_theory).  
ננסה להבין קודם כל את התאוריה ומשם נבין איך (ולמה דווקא) המספר ה8000 בבעיה שלנו מתחמק מקיומה.  
  
בתחילת המאמר, סקוט מביע את הערכתו הרבה לפרויקט מוצלח של אדם ידידיה, סטודנט לדוקטורט בMIT שהצליח לבנות מכונת טיורינג בעלת בעלת סרט אחד ושני תווים בעלת לא פחות מ-7918 מצבים, שהתנהגותה בהרצה על סרט ריק, לא ניתנת להוכחה מהאקסיומות הקלאסיות של תורת הקבוצות.

הצהרה מדויקת לתוצאה העיקרית היא,  
יש לנו מכונת טיורינג בעלת 7918 מצבים בשם Z, כך ש:  
1. Z פועלת תמידית, בהנחה שישנה עקביות של התאוריה הקרדינלית SRP.

2. לא ניתן להוכיח שZ רצה לעולם ב[ZFC](#ZFC), בהנחה שתאוריה זו עקבית (נטולת סתירות).

**רקע:**  
כתוצאה מיידית [מתאוריית אי השלמות של גֶדֶל](#תאוריית_אי_השלמות_של_גֶדֶל), נובע שישנה תכנית מחשב כלשהיא באורך מסוים, ה"חומקת" מכוחה של המתמטיקה הרגילה להוכיח מה היא עושה, כאשר היא רצה עם כמות זיכרון בלתי מוגבלת.  
לדוגמא, תכנית כזו יכלה למנות את כל ההשלכות האפשריות של אקסיומות ZFC זו אחר זו ולעצור אם תמצא סתירה כלשהיא.  
בהנחה שZFC עקבית, התכנית אמורה לרוץ לנצח, אך מצד שני ZFC **לא יכולה להוכיח** שהתכנית רצה לנצח, כי אם היא הייתה עושה זאת, היא הייתה מוכיחה את העקביות **שלה**, ומפרה בכך את משפט אי השלמות השני.  
  
לצערנו, דיון זה משאיר אותנו בחשיכה לגבי המקום בחלל תוכניות המחשב בו   
"הגרמלין הגֶדֶלי" מרים את ראשו הבלתי ניתן להכרעה.  
תכנית שתחפש אי עקביות כלשהיא בZFC לבטח תהיה יצור די מסובך.. נסביר מדוע.  
יהיה עליה לקודד לא רק את סכמת האקסיומה של ZFC, אלא בנוסף גם את השפה וכללי הסקה מהמעלה הראשונה..  
בשפת תכנות קלאסית (c, java ..), תכנית בסגנון זה עשויה להיות בעלת אלפי שורות/הוראות אם מדובר בשפת מכונה אלמנטרית.  
כנראה מעולם לא נתקלנו ולא ניתקל בתכנית כזו, גם אם היה לנו מחשב מוטרף בגודלו של היקום והיינו מריצים תכנית אחר תכנית ברנדומליות שיטתית של עכבר מעבדה.

מכאן, השאלה החשובה במקומה עומדת, שאלה שבודדים שאלו עד כה בהיסטוריה:  
האם האקסיומות של תורת הקבוצות מספיקות בכדי לנתח את ההתנהגות של כל תוכנית מחשב שאורכה לכל היותר, נניח, 50 הוראות מכונה? או שיש תוכניות סופר קצרות *שכבר מציגות "התנהגות גֶדֶלית"*?

מדעני המחשב התיאורטיים עשויים להתנגד לכך שזו "רק שאלה של קבועים".   
ובכן בסדר, כן, אבל גם מקור החיים ביקום שלנו - פאזל לא לגמרי לא קשור לנושא אך בהחלט מהותי - הוא גם "רק שאלה של קבועים"!  
ביתר פירוט, אנו יודעים שאפשר בחוקי הפיזיקה הנוכחיים שלנו לבנות מכונה המשכפלת את עצמה: לדוגמא, DNA או RNA והאביזרים הנלווים להם.  
אנו גם יודעים שמולקולות זעירות כמו H2O ו- CO2 אינן משכפלות את עצמן,  
אבל אנחנו לא יודעים **כמה קטנה** יכולה להיות המולקולה הקטנה ביותר המשכפלת את עצמה - וזה נושא שמשפיע אם עלינו לצפות *למציאת חיים כלשהם ביקום אי פעם* או שהתכבדנו להיות רק אנחנו לבד.

מה אומרים בעצם חוקי הפיזיקה או בכלל, חוקי הטבע?

מדעי הטבע הם מעשה ידי אדם, נועדו בשביל לנסות להבין את דפוסי ההתנהגות בעולמנו.

לכן, יש פיתוח מודלים בהם אנחנו לא מגלים חוקים, אלא מנסחים אותם.

נקודת ההתחלה של כל חוק מתחילה בהנחת יסוד שלא תמיד (לפעמים בהכרח) לא תואמת את המציאות, לדוגמא:

העיקרון הקוסמולוגי - קובע כי היקום כולו איזוטרופי (אין בו כיוון ייחודי ולכן יש בו סימטריה) והומוגני, אחיד בכל מקום ברמת הגלקסיות.

אך כיום ידוע שהיקום לא הומוגני כלל ועיקר, אלא כל גלקסיה עומדת בפני עצמה במרחקים עצומים זו מזו ללא אחידות כלשהיא.

אז איך בוחרים מודלים?

מצד אחד אם נהיה נאמנים מדיי למציאות - המודל יהיה מסובך ולא נצליח לבנות עליו כלום, כי לעולם לא נבין את היקום לאשורו (על המוח האנושי אנחנו יודעים מעט מאוד ומדענים תמימי דעים שחלקים נרחבים ממנו היו ויישארו בגדר חידה).

מצד שני, אם המודל יהיה פשטני , הוא לא ישקף את מה שקורה מסביבנו בצורה ראויה וטווח הטעות יהיה גדול מאוד.

**עקומת מאמץ מַעֲוָות** מתייחסת לחומר שלא קיים בטבע, ובאה להראות לנו איך לכל חומר יש גרף שונה שמייצג את הקשר בין המאמץ, הכוח שהופעל עליו, לבין השינוי שיתבצע בו.

יש חומרים שאחרי שהופעל עליהם כוח יחזרו למצב גיאומטרי זהה, כמו קפיץ

לעומת זאת, חרסינה שנפעיל עליה כוח דומה, תוכל להגיע לנקודת כניעה ומשם לנקודת שבר, אַל חָזור.

בחזרה לשאלה שלנו,  
ישנם אנשים מסוימים העשויים להתנגד לכך שמה שאנחנו שואלים עליו כבר נחקר,   
במהלך חצי המאה של המאה לעצב את מכונת טיורינג האוניברסלית הקטנה ביותר   
(היה זה נושא הפרס של סטיבן וולפרם בסך 25,000 דולר בשנת 2007).   
אבל אני רואה את זה כשונה במהותו, מהסיבה הבאה.   
למכונת טיורינג אוניברסלית - כלומר מכונה המדמה כל מכונה אחרת שמתוארת לה על קלטת הקלט שלה (לדוגמא, מכונת-טיורינג אוניברסלית U, מסוגלת "לסַמְלֶץ" חישוב של כל מ"ט M על קלט x) - יש את הפריבילגיה לפרוק כמעט את כל המורכבות שלה לפורמט התיאור של מכונת הקלט. אז כן, זה בדיוק מה שכל [המכונות האוניברסליות הזעירות](#המכונות_האוניברסליות_הזעירות) הידועות עושות! אבל לתוכנית שבודקת (נניח) את [ההשערה של גולדבך](#השערת_גולדבאך), או את [השערת רימן](#השערת_רימן), או את העקביות של תורת הקבוצות, על סרט ריק בתחילה, אין חירות כזו.   
עבור מכונות כאלה, מספר מצבים באמת עושה רושם של מדד מהותי לסיבוכיות,   
משום שהסיבוכיות הזו לא יכולה "להינעץ" בשום מקום אחר.  
אפשר לנסח את השאלה שלנו על בסיס פונקציית "הבונה העסוק" הידועה לשמצה.  
נניח ש BB(n) כלומר המספר הn-י של הבונה העסוק, מוגדר להיות מספר הצעדים המקסימלי שכל מכונת טיורינג בעלת n מצבים צועדת בריצה התחלתית על סרט ריק ובהנחה שהיא תיעצר לבסוף.  
מובן לנו ש BB(n) גדל מהר יותר מכל רצף של מספרים שלמים הניתנים לחישוב.  
איך אנחנו כל כך בטוחים בזה?  
כי אחרת, אם קצב הגדילה היה קטן/שווה, אפשר היה להשתמש בעובדה זו כדי לפתור את [בעיית העצירה](#בעיית_העצירה), בסתירה לתֵאוֹרֵמָה (הנחה הטעונה הוכחה) של טיורינג.  
  
דבר מדהים נוסף בפונקציית הBB , הוא שמדובר בפונקציה שלמה (כל פונקציה בוליאנית בינארית יכולה לבוא לידי ביטוי במונחים שלה) מוגדרת היטב, ועם זאת ברגע שנתקן את האקסיומות של המתמטיקה, ניתן יהיה **להוכיח** אי פעם רק ערכים סופיים של הפונקציה, אפילו באופן עקרוני.  
למה?  
ניקח שוב מכונת טיורינג M שתעצור אם ורק אם יש סתירה בתורת הקבוצות של ZF.  
ברור שניתן לבנות מכונה כזו עם מספר סופי של k מצבים, אך לא נוכל לחשב את הערך של BB(k) כי ZF לא תוכל לקבוע את ערכם של BB(k) (ומכאן גם BB(k+1)/BB(k+2)..) אלא כן ZF לא עקבית! בשביל זה, ZF צריכה להוכיח שM רצה לעד וכך להוכיח את העקביות שלה וכך להיות לא עקבית במשפט גֶדֶל.  
  
הגענו למצב בו נוכל לשאול שאלה פרקטית, כמותית:  
כמה ערכי פונקציית BB ניתן לדעת?  
מהי הנקודה בה הפונקציה נוטשת את תחום בני האנוש ומגיעה לעולמות עליונים?   
האם זה קרוב יותר לn=10 או בכלל לn=10,000,000?  
בפועל, נקבעו ארבעה ערכים של BB עד כה סה"כ.  
  
  
  
BB(1)=1

BB(2)=6

BB(3)=21 – לין וראדו, 1965.

BB(4)=107 (Brady 1975) – בריידי, 1975.  
  
אנו יודעים גם כמה חסמים תחתונים למספרים גבוהים יותר:  
BB(5) ≥ 47,176,870 (Marxen and Buntrock 1990)

BB(6) ≥ 7.4 × 1036,534 (Kropitz 2010)  
BB(7)>10^10^10^10^10^7.

הפסימיים מבין חובבי בעיית הBB סוברים כי BB(6) לא יהיה ידוע לעולם.  
מצד שני, הטיעון המופשט שהבאנו קודם, אומר שאם נסתפק בתורת הקבוצות של ZF, יכולים להיות עשרות מיליוני k אופציונליים, כך שכל ערכי BB(K+i) לעולם לא ניתן יהיה להוכיח. אז השאלה שלנו חוזרת: האם מספר הערכים הידועים של פונקציית BB ישאף לכמה בודדים, או יתקרב למיליון?  
  
זו השאלה בהם עסקו אדם ידידיה וד"ר סקוט ארונסון.  
אין סיכוי לתכנן מכונת טיורינג בצורה ידנית לכל המשימות הללו, כך שבתור התחלה,  
יצר אדם שפת תכנות חדשה בשם לקוניק, המיועדת לכתיבת תוכניות שמתאגדות לשפת מתווך בשם TMD ומשם לכדי מכונות טיורינג קטנות.  
  
גם לאחר כל זה, אנו מעריכים כי ניסיון ישיר לכתוב תוכנית "לקונית" שתחפש סתירה ב- ZFC יוביל למכונת טיורינג עם מיליוני מצבים.   
לכן נציג שלושה רעיונות הדרושים בכדי להוריד את כמות המצבים למספר סביר.  
  
הראשון היה להיעזר בעבודתו של הארווי פרידמן, שכתב על הבעיות האלה בעבר.  
פרידמן עמל מאז שנות השישים למצוא הצהרות אריתמטיות "טבעיות", שאינן תלויות באופן עצמאי ב- ZFC או בתיאוריות קבועות אחרות.  
הצהרותיו - הכרוכות בדרך כלל באובייקטים המכונים "גרפים בעלי סדר-משתנה" – נראות זרות ומרוחקות מקו המחשבה הסביר.  
אך כך או כך, הצהרותיו של פרידמן עדיין נראות הרבה יותר קלות לקידוד כתוכנות מחשב קצרות מאשר המנגנון המלא של הלוגיקה ותורת הקבוצות מהסדר הראשון!  
  
הרעיון השני היה משהו שקראנו לו "עיבוד בסרט".  
בעיקרון, במקום לקמפל ישירות מ- Laconic מטה עד למכונת טיורינג,  
אדם כתב מתורגמן ([interpreter](#interpreter)) במכונת טיורינג (שלקחה כ -4000 מצבים - עלות קבועה יחידה), ואז מכונת טיורינג הסופית כתבה תחילה תוכנית ברמה גבוהה יותר לסרט שלה ופירשה אותו לאחר מכן. במקום שתהליך ההידור מייצר תְּקוּרָה עצומה במספר מצבי מכונות טיורינג (ומכונה חוזרת), גישה זו נותנת לנו תקורה נוספת בלבד.   
נמצא שרעיון זה הפחית את מספר המצבים לאין שיעור.  
  
הרעיון השלישי הוצע לראשונה בשנת 2002 ואנו קוראים לזה "קידוד אינטרוספקטיבי (מביט פנימה)". כאשר אנו כותבים את התוכנית שתתפרש על קלטת מכונת טיורינג, הגישה הנאיבית תשתמש במצב מכונת טיורינג אחד לכל ביט. אבל זה לחלוטין בזבזני, מכיוון שבמכונת טיורינג בעלת n מצבים, כל מצב מכיל log (n) ביטים של מידע (בגלל המצבים האחרים שהוא צריך להצביע עליהם).   
גישה טובה יותר מזו מנסה לנצל כמה שיותר מאותם ביטים.  
פעולה זו מקנה לנו חיסכון נוסף של עד פי 5 במספר המצבים.

עבור השערות גולדבך ורימן, "שילמנו" את אותה תקורה של 4000 מצבים עבור המתורגמן, אבל אז התוכנית שצריך היה לפרש הייתה פשוטה יותר, ונתנה מכונה כוללת קטנה יותר. אגב, לא ברור באופן אינטואיטיבי שהשערת רימן מקבילה לאמירה שתוכנית מחשב מסוימת פועלת לנצח, אך היא נובעת מעבודות קודמות.

כדי להקדים את השאלה הבלתי נמנעת,   
כן, הרי שהפעלנו את מכונות הטיורינג האלה לזמן מה, ולא, אף אחת מהן לא נעצרה אחרי יום בערך. אבל לפני שנפרש את זה כראיה לטובת גולדבך, רימן והעקביות של ZFC,   
אתה כנראה צריך לדעת שמכונת טיורינג הבודקת אם כל הריבועים המושלמים הם פחות מ -5, המיוצרת באמצעות Laconic, צריכה לרוץ יותר מ- שעה(!) לפני שמצא את הדוגמה הנגדית הראשונה (כלומר 3^2=9) ונעצרת.   
מכונות טיורינג ע"י Laconic מותאמות רק למספר המצבים, לא למהירות ריצה, בלשון המעטה.

כמובן, עדיין נותרו שלושה סדרי גודל בין הערך הגדול ביותר של n (4) שעבורו BB (n) ידוע במתמטיקה מבוססת ZFC, לבין הערך הקטן ביותר של n (7,918) שעבורו ידוע כי BB (n) אינו ניתן לחישוב. יש אופטימיות לשיפורים נוספים במכונה Z - בין אם זה אומר הפשטה של הצהרת פרידמן, מתורגמן שעוצב מחדש (אולי באמצעות חישוב lambda?),   
או "מודל רקטות רב שלבי" שבו מתורגמן חשוף ופשוט מאוד, יפרק מתורגמן שני עשיר יותר, שישמש לפירוק שלישי וכו', עד שתגיע לתוכנית שאכפת לך ממנה.   
זה יהיה לא פחות מהלם אם מישהו בעשרות שנים הקרובות יקבע את הערך של BB(10) למשל, או יוכיח את הערך הבלתי תלוי בתורת הקבוצות.  
גם לאחר שהסינגולריות מתרחשת, סביר להניח שהקביעה של BB(10) תישאר די מאתגרת.

# "Attacking the Busy Beaver 5"

**תמצית מאמר:**מאז שהגדיר ראדו בשנת 1962 את משחק הבונה העסוק, השתמשו במספר גישות

טכנולוגיות לחיפוש בונים עסוקים או אפילו לחשב ערכים מיוחדים של Σ.  
תזכורת: Σ = הכמות המרבית של סימנים שהמכונה יכולה להדפיס לפני שהיא עוצרת

Σ(5) עדיין לא ידוע. נציג גישה חדשה לחישוב Σ(5),  
יחד עם תוצאות ראשוניות, במיוחד Σ(5) ≥4098.   
גישה זו כוללת טכניקות להפחתת מספר מכונות הטיורינג הנבדקות, בכדי להאיץ הדמיה של מכונות טיורינג ולהחליט על אי -הפסקתם.  
  
**פְּרֶה מאמר:**משחק הבונה העסוק מבקש לבנות מכונות טיורינג פשוטות המייצרות כמות מקסימלית של אחדות על הסרט שלהן לפני עצירתן. למכונות טיורינג אלה יש N מצבים, (אחד מהם הוא המצב ההתחלתי), אלפבית הקלט עם שני תווים, בד"כ 0 ו- 1, סרט ראשוני עם 0 סמלים, מצב עצירה אנונימי, ובכל שלב נדפיס תו חדש ונזיז גם את הראש הקורא-כותב ימינה או שמאלה לאורך הסרט שאינו מוגבל באורכו.  
עבור כל Nהמצבים, יש מספר סופי של מכונות טיורינג כאלה.  
Σ(N) היא פונקציה המוגדרת להיות המספר המקסימלי של אחדות שמכונת טיורינג כזו עם N מצבים מייצרת על הקלטת שלה עד לעצירתה.   
מכונת הטיורינג שתייצר לנו את Σ(N) תיקרא – הבונה העסוק.

הוּכַח ש- Σ אינה [פונקציה רקורסיבית כללית](#general_recursive_function), מכאן שאינה ניתנת לחישוב.   
Σ (1) = 1 הוא טריוויאלי, Σ (2) = 4 קל להראות, את Σ(3)=6 ו Σ(4)=13 כבר הראו,

אך (5) Σ עדיין לא ידוע. לאט לאט הציבו לו חסמים תחתונים גדולים יותר ויותר.  
בתחילה זה היה Σ(5)≥501, ב1984 נמצא כי Σ(5)≥1915 ובאוגוסט 1989 החסם עלה ל .Σ(5)≥4098

בעוד שניתן להציג חסם תחתון של Σ(N) עבור N מצבים מסוים על ידי מכונת טיורינג אחת, חישוב שלΣ(N) עצמו דורש **להוכיח** שכל מכונות הטיורינג האחרות של N מצבים אינן נעצרות או לא מייצרות יותר אחדות.

ברור שבעיית העצירה של מכונות טיורינג (שכידוע, אינה ניתנת לחישוב) מעורבת, ומסבכת את העניינים.  
  
**האלגוריתם הכללי:**  
כמו חוקרי העבר האחרים, אנו עוקבים ובונים באופן ישיר אחר ההגדרה של Σ(N),   
''המספר המקסימלי של אחדות המיוצר על ידי מכונת טיורינג עוצרת (מהסוג המתאים שהגדרנו) בעלת N מצבים''

לפיכך האלגוריתם הכללי הוא:

1. ספור את כל מכונות הטיורינג M בעלות N המצבים, ועבור כל אחת מהן (2).

2. סַמְלֶץ את ההתנהגות של M. התחל עם סרט ריק, עד ש (3), (4), (5 א') או (5 ב') יקרה.

3. M עוצר. ספרו את האחדות בסרט המכונה, השוו לשיא הנוכחי ובמקרה הצורך עַדְכְּנוּ אותו.

4. M הוּכַח כמכונת טיורינג שלעולם לא תיעצר. תשכח מM.

5 א'. גודל הסרט חורג ממגבלה מסוימת שהוגדרה מראש.  
סַמֵּן את M בהערה כמכונה שאין הסכם לגביה (undecided).

עם ייצוג הולם של הסרט, זה יקרה לעיתים נדירות.

5 ב'. מספר שלבי הסימולציה חורג ממגבלה מוגדרת מראש.  
סַמֵּן את M בהערה כמכונה שאין הסכם לגביה (undecided)

גישות קודמות עזבו במקרה זה (4). אחרות ניסו מימושים מוגבלים.   
בניגוד לכך אנו מדגישים שיטות יעילות להחלטה על אי-עצירה.  
ניתן למצוא בונים עסוקים ללא (4), החישוב של Σ צריך את זה.

על מנת להשתמש באלגוריתם הנ"ל לחישוב Σ(5) בתוך מגבלות הזמן המעשיות (כחודש לדוגמא), יש להשיג את שלושת תתי המטרות הבאות:

* 1. לא צריך לקחת בחשבון את כל מכונות הטיורינג האפשריות קומבינטורית.  
     יש לנו חסם עליון של N2(6N), זו הפסגה.  
     זה יוצא 146\*10 עבור N = 5, לבטח הרבה יותר מדיי.   
     גישה משופרת יותר הציעה N2(4N+4) שלטובת החישוב של Σ יכולה להשתפר ל N-22(4N)\*(2N-1). הפעם, עבור N=5, זה יוצא 2.3\*1011, עדיין גדול מדיי. כותבי התכניות כרגע מנו פחות מ9\*107 מכונות טיורינג.
  2. האצת סימולציית ההתנהגות של מכונות טיורינג.   
     הבונה העסוק הנוכחי בעל 5 המצבים עושה למעלה מ-4.7\*107 צעדים לפני שהוא עוצר. זה כבר גדול מספיק בכדי למשוך תשומת לב מיוחדת.
  3. ההכרה בכך שלעולם לא עוצרים מכונות טיורינג צריכה להיות יעילה, כמו כן לקרות מוקדם בתהליך של סימולציה וצריכה להיות שלמה ככל האפשר והכרחית עבור N=5.  
       
     בפרקים הבאים נתאר איך להשיג את שלוש המטרות הללו.  
       
     **ספירת מכונות טיורינג:**

גם במקרה אוטופי בו יכלה להיות החלטה מיידית עבור כל מכונת טיורינג,   
ספירת כל המכונות האפשריות בצורה קומבינטורית אינה מעשית כלל וכלל.

המספרים כבר גדולים עד כדי מניעה כבר עבור N = 5.  
למזלנו, זה לא קשה מאוד לצמצם את המספרים הללו באופן משמעותי.

אם יש קבוצת S של מכונות טיורינג כאלה שאף אחת מהן לא עוצרת, או שכולן עוצרות ומייצרות את אותו מספר אחדות, אזי - על מנת לחשב את Σ - מספיק להחליט/לסַמְלֶץ רק מכונה אחת מתוך S. המטרה היא לבדוק אם ניתן לסדר את S בצורה כזו שהאלמנט הקטן ביותר בה ייצג אותה.

ערכות שקילות כאלה, הן המנגנון הבסיסי להפחתת כמות הספירה והחישוב.

מחלקת שקילות חשובה ביותר כבר שימשה את החוקרים בעבר, שם התוצאה שלה אפיינה עץ (TNF). זה מזהה את קבוצת המכונות השונות אך ורק בשמות המצבים (איזומורפיזם)

ובמעברים שמעולם לא נעשו. כל המכונות הללו מפגינות ללא ספק התנהגות זהה.   
לפני הסימולציה בדרך כלל לא ידוע איזה מעבר יבוצע, לכן אנו מגיעים בדרך הבאה

כדי למנות את כל מכונות הטיורינג על ידי ספירת מעברים בודדים במהלך סימולציה:

1. סַמֵּן את כל N2 המעברים הלא מוגדרים של המכונה M לא מוגדרים. בצע את שלב 2

2. סַמְלֶץ את M (המייצג מחלקת מ"ט) עד שאחד מ 3, 4, 5 או 6 יקרה.

3. M עצרה. ספור את כמות האחדות.

4. M הוּכַח, לעולם לא תעצור. תשכח מM.

5. גודל הסרט/כמות הצעדים חורג ממגבלה מסוימת שהוגדרה מראש.  
סַמֵּן את M בהערה כמכונה שאין הסכם לגביה (undecided)

6. בֻּצַּע שימוש במעבר המסומן כבלתי מוגדר. סְפוֹר את כל הערכים המשמעותיים למעבר, ובצע את שלב (2) רקורסיבית עם הקבוצה M לכל אחת מהמכונות השלמות והמוגדרות יותר.

מכלול ערכי המעבר החוקיים נוצר על ידי שילוב שרירותי של N+1 מצבי מטרה, 2 סמלים לכתוב, ושני כיוונים לזוז.   
קבוצת המעברים המשמעותיים המנויים בשלב (6) היא תת קבוצת של כל

החוקיים לפי כללי הרֵדוּקְצִיָּה הבאים:

• מקבוצת המצבים M שעדיין לא נלקחו, רק הקטן ביותר (על פי ברירת מחדל שתהיה

בעלת סדר קבוע על המצבים) משמש כמצב מטרה (איזומורפיזם).

• בעת הגדרת המעבר הראשון שמכריח שינוי מצב (אחרת אף פעם לא נעצור), כתוב 1

(איזומורפיזם), וזוז שמאלה (סימטריה). זה לחלוטין יתקן את המעבר הראשון.

• בעת הגדרת המעבר האחרון, שקול **רק** את מצב העצירה כמצב יעד.

• אם מצב היעד הוא מצב העצירה, לך שמאלה וכתוב 1. זֶה יתקן לחלוטין את המעבר האחרון.

• אם יש מצב ש- M עדיין לא לקח, אל תחשיב את מצב העצירה כמצב היעד

(Σ (N+1)> Σ (N)).

זה בערך מסכם את ה- TNF. ישנם עוד מערכות שקילות בעלי חשיבות בינונית שלא הכרחי להתעכב עליהן.  
  
**האצת הסימולציה:**

טכניקת ההאצה הבסיסית היא שימוש במכונות מאקרו.  
K מכונות מאקרו פועלות על בלוקים של K תווי סרט (ביטים במקרה שלנו) כתו מאקרו אחד. יש לו קבוצת מצבים משוכפלת לקודד בין אם השמאלי או החלק הימני ביותר של סמל המאקרו נמצא מתחת לראש הקורא-כותב.  
למכונת K-macro עבור מכונת טיורינג בעלת N מצבים יש K2\*N\*2 מעברים,

שכל אחד מהם מוגדר על ידי הרצת המכונה המקורית על סרט מוגבל של

באורך K.

טכניקת ההאצה הבאה והחשובה ביותר היא שימוש בספירות חוזרות ומפורשות של תווי הסרט. לדוגמא: abbaaab יקודד ל- a1b2a3b1.

ההאצה הברורה ביותר מתרחשת כאשר צעד אחד מכסה כמות גדולה של תווים.  
ככלל אצבע, אם מכונת טיורינג מבצעת צעדים רבים לפני עצירתה, היא לבטח מבצעת איטרציות פשוטות רבות מסוג זה.

**החלטות אי סיום:**

ההחלטה שמכונת טיורינג לעולם לא תעצור נעשית תמיד על ידי חיזוי התנהגות המכונה

בדיוק מספיק, בו נהיה בטוחים שלעולם לא תשתמש במעבר שטרם הוגדר.  
כלומר, מצבים שכבר מוגדרים יהיו בשימוש חוזר אינסופי.   
ניתן להגיע להחלטה זאת באמצעות טיעונים שונים למדי:

א. הראה באופן קונסטרוקטיבי (בונה) את דפוס ההתנהגות הבא (forward reasoning).

ב. הראה שכל הנתיבים שאליהם מגיעים באמצעות מעבר אחר אינם יכולים להתחיל כאן (backward reasoning).

עבור ב': כאשר בתהליך הספירה יוגדר מעבר למצב עצירה, הוא יתבצע באופן מיידי,

המכונה תוּערך ותוּשׁלָך. לפיכך, במהלך סימולציה רגילה, כל המעברים המוגדרים

אינם עוצרים. נגדיר D ו- U להיות קבוצת המעברים המוגדרים ושטרם מוגדרים.   
כדי לעצור, יש צורך במעבר אחד u ∈U שיהיה נחוץ ומוגדר.  
יש דרך אקסקלוסיבית להגעה אליו, דרך מעברים מD.  
כלומר, אחד מתוך קבוצת P של כל הנתיבי באורך L (L≤S+1) מהצורה DL-1 × U, כאשר

S הוא מספר הצעדים שכבר נעשו על ידי המכונה המדוברת.  
נגדיר Q להיות קבוצת כל אותם p ∈ P, האפשריים על פי שתי ההגבלות.   
ראשית, מצב היעד של כל מעבר ב- p חייב להתאים למצב המקור של המעבר הבא בp. שנית, עבור כל מעבר t בp, הֶקשֵׁר הסרט שהוגדר על ידי המעברים ב- p לפני t מחויב שלא לסתור את סמל המקור של t.  
  
אם יש L כזה ש- Q הוא ריק, אז כל u אינם ניתנים להגעה והמכונה לא יכולה לעצור יותר.

עבור א': אם נמצא דפוס פעולה עתידי שבו אנו יכולים להוכיח שהוא יחזור על עצמו אינסופית, המכונה לעולם לא תעצור. זו עדיין אמירה כללית מאוד. השלבים הם:

• אם יש קבוצת מצבים S כך שכל המעברים מרכיבי S מוגדרים ומצב היעד שלהם

נמצא גם ב- S, (והמכונה נמצאת באחד המצבים הללו) היא לעולם לא תעזוב שוב את S ובכך לא תיעצר.

• אם מכונה משחזרת את אותה תצורה/קונפיגורציה בדיוק (מצב וסרט ביחס לראש שלה), היא תחזור על כך בדיוק.

• אם מכונה מייצרת ''תצורה גדולה יותר'' היא תמשיך בסופו של דבר לייצר אפילו יותר

תצורות (שימוש חוזר באותם מעברים).

**כיוונים עתידיים:**

היה צורך במאמץ ניכר בכדי 'לתקוף' Σ(5) וזה עדיין לא מספיק.  
גם כמות זמן החישוב אינה טריוויאלית (כעשרה ימים באמצעות מעבד Clipper 33 מגה-הרץ).  
כמות האופציות הקומבינטוריות בגבולות העליונים מצביעות על כך שהחישוב של (6)Σ יצטרך בערך פי 500 פעמים מכונות רבות יותר..

עם השנים וההתקדמות הטכנולוגית לצידן, ניתן לבצע חלוקת עבודה בקלות רבה יותר, ועם מספר משוער של כ-2000 מחשבים (תחנות עבודה), נראה שעדיין ניתן לתקוף (6)Σ.

# הוכחת אי כריעות בעיית busy beaver מתוך הוכחת halting problem

הגדרת מכונת טיורינג היא חזקה, כלומר מוגדרת היטב:  
שינוי האלפבית, מס' הסרטים וכו' -> לא ישנו את הכוח החישובי שלה.  
לכן, מבלי הגבלת הכלליות נניח שמכונת טיורינג M מורכבת מ:  
אלפבית Γ

קבוצת מצבים Q  
פונקציית מעבר δ: Q × Γ כך ש:  
Q × Γ →3

{L, S, R}×2

2

ושלושה סרטים: קלט, עבודה, ופלט.  
  
הגדרה 1 – מכונת טיורינג אוניברסלית  
\* למכונת טיורינג קלאסית: (Γ; Q; δ)=<M> ניתן להתייחס כאל מחרוזת {0, 1}sdeeddwdDd∗.  
\* מכונת טיורינג u תיקרא אוניברסלית אם U(<M>,x) = M(x),  
 כלומר, מדובר במכונה המסוגלת "לסמלץ" חישוב של **כל** מ"ט M על קלט x.  
  
נשים לב כי המהדר הוא בגודל **קבוע**, אך הוא יכול לקמפל קוד **בכל** **אורך** שיקבל.

הקלט ל- U הוא התיאור של TM, למשל: Γ||Q||δ||x   
("||" מייצג שרשור בצורה שאינה משתמעת לשתי פנים),  
U מדמה את הקלט TM.

נוכל לקבל מֻסְכָּמָה C בנוגע לפירוש מחרוזת שרירותית ב- \*{0, 1} כמ"ט.

למשל, אם מחרוזת לא הגיונית במֻסְכָּמָה C שצוינה מראש,  
אנו מתייחסים אליו כאל TM מיוחד שתמיד לא יפלוט דבר ויעצור מיד לאחר התחלתו.

C אינו ממפה כל מחרוזת ב- \*{0, 1} לכדי מ"ט בעלת משמעות **בדיוק** באותו אופן בו בעולם האמיתי חלק מהתוכניות אינן מתקמפלות.  
  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
  
לאחר מכן, כיצד נבנה TM u אוניברסלי?  
  
תיאור של u:  
 uהוא מתורגמן. יש לו שני חלקים של קלט M המקודדים כ- <M> וקלט x המוזן ל-M.   
u משתמש בקלטת עבודה כדי לעקוב אחר תצורת זמן הריצה של M .  
u מדמה כל שלב של M(x).

בכל שלב כזה, u סורק את התצורה הנוכחית של M בסרט העבודה של u וסורק את פונקציית המעבר ב- <M> כדי למצוא את הפעולה בהתאם.  
u מבצע את הפעולה המתאימה על ידי עדכון התצורה.

לבסוף, פלט של M (x) נכתב לסרט הפלט.  
  
תֵּאוֹרֶמָה 2: (קיום של u)  
מכונת טיורינג האוניברסלית קיימת.

**בעיית העצירה:**  
הגדרה 3 (בעיית העצירה)  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
אינטואיציה: אתה יכול להוכיח אם תוכנית ספציפית (לדוג', תוכנית למציאת

הנתיב הקצר ביותר) תעצור או לא בסופו של דבר.   
אך אינך יכול לכתוב תוכנית "**אוניברסלית**" המתאימה **לכל** התוכניות.

זה גורר לכך שאין אוֹטוֹמַצְיָה (תהליך בו מכונה תחליף בן אדם) להחלטה על עצירת TM.

תֵּאוֹרֶמָה 4: (halt אינה ניתנת לחישוב uncomputable)  
הוכחה:  
נניח בשלילה כי halt ניתנת לחישוב ע"י מ"ט M0 ונגדיר:  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
אם Halt ניתנת לחישוב אז גם הפונקציה g. נניח ש- g ניתנת לחישוב על ידי מ"ט M.  
מהו g(<M>)?  
  
\* אם g(<M>)=0, אזי לפי הגדרת g ( ), Halt(<M>, <M>)=1, מה שאומר ש M(<M>) תהיה בלולאה אינסופית. לכן, g(<M>) **≠** M(<M>).  
  
\* אם g(<M>) בלולאה אינסופית, אז לפי הגדרת g ( ), Halt(<M>, <M>)=0, מה שאומר   
ש- M(<M>)ייעצר. לכן, g(<M>) **≠** M(<M>).

הערה:  
ההוכחה שראינו היא בשיטה שנקראת אלכסוניזציה. היא הומצאה על ידי גיאורג קנטור בכדי להוכיח שהקרדינליות (העוצמה) של מספרים אמיתיים גדולה מזו של מספרים רציונליים .

הגדרה 5: (הבונה העסוק)  
ניקח בחשבון מ"ט עם סרט אחד בלבד.  
 האלפבית Γ={0,1}. אין לנו קלט. הסרט התחלתי הוא 0 . . .00.

אם הם נעצרים, יוצג תוכן הסרט כפלט.   
המספר המקסימלי של האחדות בסרט הקלט לאחר עצירת מ"ט במספר n-י של מצביםBB (n) =

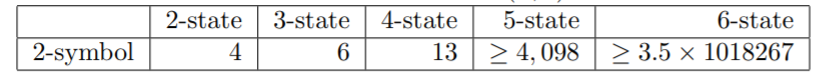
def = מספר מקסימלי של 1 בפלט הקלטת

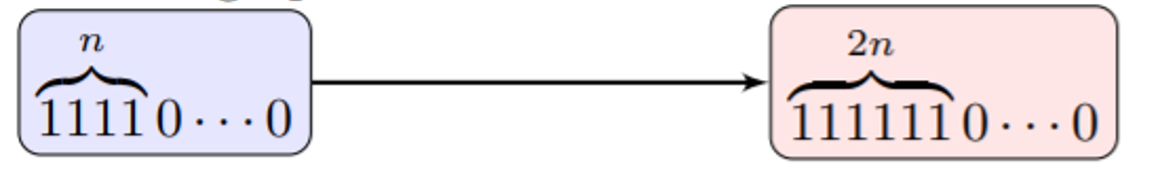
על ידי עצירת TM עם n מצבים  
  
הערה היסטורית:  
בונה עסוק הוא מ"ט שמשיגה את מספר הצעדים המרבי המבוצע **או** מספר הסמלים הלא ריקים המתקבלים בסיום הריצה על סרט הפלט בקרב כל המ"ט במחלקה מסוימת כלשהיא.  
המ"ט במחלקה זו חייב לעמוד במפרטי עיצוב מסוימים ונדרש בסופו של דבר לעצור לאחר שהתחיל עם סרט ריק.

פונקציית הבונה העסוק מכמתת את הגבולות העליונים הללו לפי אַמַּת מִדָּה נתונה, והיא פונקציה שאינה ניתנת לחישוב.

למעשה, ניתן להראות כי פונקציית בונה עסוקה גְּדֵֵלָה מהר יותר באופן אסימפטוטי מכל פונקציה אחרת הניתנת לחישוב.

הרעיון הוצג לראשונה על ידי טיבור ראדו. לדוגמה, תנו ל- Σ (n, 2) לציין את המספר הגדול ביותר של אחדות הניתן להדפסה ע"י מכונה בעלת n מצבים ושני תווים (כרגיל, {0,1}).  
המ"ט תתחיל על סרט ריק לפני עצירתה.   
המיצג הבא נותן תחושה כמה מהר פונקציה זו מזנקת

**טבלת ערכים של Σ (n, 2):**

הוכחה:  
נניח בשלילה שBB ניתנת לחישוב ע"י מ"ט M בעלת k המצבים.  
ניתן לDBL להיות מ"ט בעלת k מצבים שמבצעת את האופרציה הבאה:  


נגדיר את n להיות גדול מספיק בכדי להיות ספציפי מאוחר יותר.  
שקול את מכונת הטיורינג הבאה Mn:  
\* כתוב n אחדות לסרט: תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
\* אַפְשֵׁר ביצוע DBL: תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
\* אַפְשֵׁר M: תמונה שמכילה טקסט, שעון

התיאור נוצר באופן אוטומטי  
\* אַפְשֵׁר M פעם נוספת תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
  
  
  
כמה מצבים יש ל- Mn?  
השלב הראשון לקח n מצבים; השניk מצבים, והשלישי&רביעי k מצבים.  
לכן, סה"כ נקבל תמונה שמכילה טקסט, שעון

התיאור נוצר באופן אוטומטימצבים. נקבל ש:  
  
נשים לב שBB היא פונקציה מונוטונית עולה כיוון ש- BB(n+1) ≥ BB(n) **לכל** n טבעי.  
לכן BB(2n) ≤ n+2k+k הוא לחלוטין בלתי אפשרי לn גדול מספיק (לדוגמא, n>2k+k). 

דוגמאות הרצה busy beaver  
תיאור התחלתי:  
המצבים שנתאר בהם את המכונה יכולים להיות כקובצי טקסט הנטענים ואז מתפרשים על ידי מכונת טיורינג. נשתמש בסימון סטנדרטי לתיאור המצבים:

(מספר מצב, התו שנקרא) -> (מספר המצב הבא, תו לכתיבה) תא הבא.

כל מצב מורכב בדרך כלל משלושה כללים, אחד לכל אחד משלושת התווים (0, 1, ריק) שאפשריים להיקרא מכל תא. אז הכלל הראשון במדגם המצבים הבא אומר למכונה:

אם המכונה נמצאת במצב "1" ויש אפס בתא, שני אותו לאחד, שנה למצב "0" ולך תא אחד שמאלה.

(0,1)<-(1,0) שמאל

(1,0) <- (1,1) שמאל

(0, 1) <- (B,1) ימין

הכלל השני אומר למכונה:

אם יש אחד בתא, שנה אותו זה לאפס, השאר את המצב כ- "1" ולך תא אחד שמאלה.

החוק השלישי אומר למכונה:

אם התא ריק, שנה אותו לתו אחד, עבור למצב "0" ולך תא אחד ימינה.

הכלל הנוסף היחיד שנראה בתוכניות הוא מהלך התא הבא של "עצור" (halt).

(0, 1) <- (B,1) עצירה

זה בדיוק כמו שזה נשמע, כאשר פועלים לפי חוק זה התא משתנה מblank לאחד והמכונה עוצרת.

לסיכום:

הכללים הפועלים במכונת טיורינג הם פשוטים. שינוי האפסים, העברת תא אחד לשמאל או לימין, מושגים אלה הם קלאסיים, אך הם יכולים לחשב **כל** **דבר** שניתן לחשב אותו.  
מהמושגים הפשוטים הללו נולדים המחשבים המורכבים ביותר של ימינו.

A 4-State Busy Beaver

בעיית הבונה העסוק היא, כפי שראינו, בעיה תיאורטית מעניינת במדעי המחשב.  
באופן פורמלי זה הולך בערך כך - בהינתן מכונת טיורינג בעלת n מצבים עם אלפבית של שני סימנים {0, 1}, וסרט התחלתי ריק,   
מהו המספר המרבי של אחדות שהמכונה עשויה להדפיס על הסרט לפני שהיא עוצרת?

בעיה זו מתבררת כבלתי ניתנת לחישוב, כלומר למספר מצבים קטן ניתן למצוא תשובה, אך באופן כללי לא ניתן לפתור אותה. תיאורטיקנים מכנים בעיות כאלה שאינן ניתנות לחישוב. מספר הצעדים הנדרשים כדי להריץ עד לסיום (עצירה) גדל במהירות רבה ככל שמספר המצבים גדל.  
כך, לשלושה מצבים לוקח למ"ט 14 צעדים ואילו דוגמה זו שנראה מטה של 4 מצבים, כבר לוקחת 107 צעדים. ככל שנוסיף מצבים, נעלה בצורה דרסטית אל הלא נודע.

ימין (0,0) -> (1,1)

שמאל (0,1) -> (1,1)

ימין (0,B) -> (1,1)

שמאל (1,0) -> (0,1)

שמאל (1,1) -> (2,0)

שמאל (1,B) -> (0,1)

עצירה (2,0) -> (2,1)

שמאל (2,1) -> (3,1)

עצירה (2,B) -> (2,1)

ימין (3,0) -> (3,1)

ימין (3,1) -> (0,0)

ימין (3,B) -> (3,1)

# מושגים:

Zermelo-Frankel Set Theory  
תאוריה מתמטית בעלת 8 אקסיומות, שנועדה לפרמל מושג פרימיטיבי אחד,   
של קבוצה סופית שהאלמנטים בה סופיים והיא well-founded (מורכבת ממספר סופי של קבוצות מקוננות בלבד), כך שכל הישויות ביקום זה הן קבוצות כאלה.  
תורה זו הביאה לקיומו של יקום תיאורטי-קבוצתי כה עשיר עד שניתן לפרש את **כל** האובייקטים המתמטיים כסטים.  
היא משמשת **כבסיס** לכל המתמטיקה בזכות האקסיומות שלה.

ZFC  
הרחבה לZF יחד האקסיומה התשיעית של הבחירה ( C מייצג "בחירה", choice), והיא אומרת שלכל משפחה לא ריקה של קבוצות, יש את פונקציית הבחירה שמחזירה פריט רנדומלי מהקבוצה.

האם ZFC עקבית?  
שאלה ענקית. אנחנו **מאמינים** שכן, אך **לא** **נוכל** **להוכיח** זאת לעולם הודות לתאוריית אי השלמות של גֶדֶל. אפשר לטעון שלא מצאנו סתירה, אך מצד שני בדקנו 0% מהמקרים כי יש אינסוף מקרים בZFC. אז מה מניח את דעתנו בהקשר לאמונה הכמעט עיוורת הזו?  
  
התשובה לכך גובלת בפילוסופיה.  
נדמה את השאלה הזו להימצאותם של חדי קרן ורודים בעולם אינסופי.  
האם כל יום שעובר בו איננו מוצאים כאלה ביקום חסר הגבולות שלנו, הוא לא סוג של הוכחה לאי קיומם?

בואו נעשה את זה רשמי יותר.

בעוד שבמובן מסוים האקסיומות המתמטיות (והתיאוריות הנובעות מהן) אינן מוגבלות,   
איננו בוחרים מהן באופן אקראי בבחירת מה ללמוד.   
ZFC קמה מהאינטואיציה שלנו לגבי קבוצות שנבעו מעבודה עם קבוצות ומספרים במשך אלפי שנים. אז המערכות שאנו בוחרים לעבוד איתן במובן לא טכני כלשהן "פחות סבירות" להיות לא עקביות מאשר אם רק יצרנו את האקראי שלהם באופן אקראי.

כמו כן, אנו נוטים לגלות חוסר עקביות די מהר ברגע שנתחיל בניתוח רשמי של מערכות. לדוגמא, הפרדוקס של ראסל נמצא עוד לפני שגוטלוב פרגה סיים להוציא את ספרו.

גם אם ZFC אינו עקבי בצורה עדינה כלשהי, זה היה כל כך מוצלח כל כך הרבה זמן עד שכנראה ניתן למצוא מערכת עקבית דומה ללא יותר מדי אובדן כוח.

לבסוף, תיאורטיקנים של תורת הקבוצות השקיעו עשרות שנים בחקר תיאוריות חזקות עוד יותר שלא נמצאו כלא עקביות. אם הייתה חוסר עקביות ב- ZFC, סביר להניח שהוא יעלה מוקדם יותר ברגע שתתחיל להוסיף אקסיומות חזקות נוספות כמו עוצמות גדולות יותר.

לבסוף, **זו** רק הסיבה לכך שתאורטיקנים מאמינים ש- ZFC עקבי. עבור המתמטיקאי הממוצע, שאלת השאלה הזו היא כמו לשאול "מדוע אתה מאמין שישנו יקום עקבי ואובייקטיבי, ולא כזה שנוצר לפני אלפית השנייה בצורה זו בדיוק וכל חוקי הפיזיקה פשוט יפסיקו לפעול מחר?" פילוסופיה לחוד, התגובה של רוב האנשים מסתכמת ב"כיוון שהיא פשוטה ושימושית יותר לחיי אם אני לא דואג ברצינות לאפשרות הזו, וגם אם היא הייתה נכונה, היסודות הפילוסופיים של העולם שלי הם פחות הדאגות שלי.."

מכונות אוניברסליות זעירות:  
כשאלן טיורינג העלה את הרעיון של מכונה אוניברסלית, הוא חשב על מודל המחשוב הפשוט ביותר שיהיה חזק דיו בכדי לחשב את כל הפונקציות האפשריות שניתנות לחישוב.  
היה זה קלוד שאנון שבשנת 1956 העלה לראשונה באופן מפורש את שאלת מציאת מכונת טיורינג האוניברסלית הקטנה ביותר האפשרית.  
הוא הראה ששני תווים (בד"כ 0 ו-1) מספיקים כל עוד משתמשים במצבים מספיקים   
(או להפך), ותמיד ניתן להחליף מצבים בתווים הללו.  
 הוא גם הראה שאף לא מכונת טיורינג אוניברסלית אחת של מצב בודד יכולה להתקיים.

בעיית העצירה:

מהמרכזיות בבעיות בתחום החישוביות.  
מנוסחת כבעיית ההכרעה (כן/לא) כדלקמן:  
בהינתן תוכנית מחשב וקלט, האם התוכנית תסיים את פעולתה בשלב כלשהו עבור קלט זה?  
  
תאוריית אי השלמות של גֶדֶל  
לפי משפט השלמות של גודל, אם ZF עקבי, אז הוא מספק, ולכן יש קבוצה V אוספת את כל המערכות ביקום של ZF, שנראית סתירה.  
  
interpreter  
מְפָרֵשׁ, הוא תוכנה הקוראת תוכנית מחשב הכתובה בשפת תכנות ומבצע אותה ישירות, פקודה אחר פקודה.

השערת גולדבך

השערה בתורת המספרים, שלפיה כל מספר זוגי גדול מ-2 ניתן להציג כסכום של שני מספרים ראשוניים.

השערת רימן

החלק הממשי של כל ה[אפסים](https://he.wikipedia.org/wiki/%D7%90%D7%A4%D7%A1_%D7%A9%D7%9C_%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%94) (ה[לא טריוויאליים](https://he.wikipedia.org/wiki/%D7%98%D7%A8%D7%99%D7%95%D7%95%D7%99%D7%90%D7%9C%D7%99_(%D7%9E%D7%AA%D7%9E%D7%98%D7%99%D7%A7%D7%94)) (של [פונקציה מרוכבת](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%94_%D7%9E%D7%A8%D7%95%D7%9B%D7%91%D7%AA) הידועה בשם **"**[פונקציית זטא של רימן](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%99%D7%AA_%D7%96%D7%98%D7%90_%D7%A9%D7%9C_%D7%A8%D7%99%D7%9E%D7%9F)" הוא .

פונקציה רקורסיבית כללית (general recursive function)  
ישנם שני מחנות מחשבה על המשמעות של פונקציה רקורסיבית כללית.  
מחנה אחד רואה בפונקציות רקורסיביות כלליות שקולות לפונקציות הרקורסיביות הרגילות. עבור חברי מחנה זה, המילה "כללית" מדגישה כי סוג הפונקציות כולל את כל תת -המעמדות הספציפיים, כגון הפונקציות הרסיביות הפרימיטיביות (רוג'רס 1987, עמ '27).

המחנה השני רואה בפונקציות רקורסיביות כלליות שוות ערך לתפקודי רקורסיביות הכוללים.

# כיווני מחקר עתידיים:

ננסה לחשוב על אפיקי חשיבה יצירתיים לאחר שנקבל בסיס מוצק של הבעיה בתוספת מאמרי אקדמיה שהרחיבו את היריעה עליה.

ראינו שמכונת טיורינג היא הפשטה של המחשב **הדיגיטלי**.  
 היא סוג של מחשב שהקשר שלו לעולם מתבצע באמצעות סרט עליו כתוב הקלט, שעליו נכתב הפלט.   
הסרט מחולק לתאים, בכל תא כתוב סימן בודד – (מקובל להשתמש בסימנים 0 ו-1).   
למחשב ראש קורא/כותב שנמצא בכל רגע נתון מעל אחד התאים שבסרט.

למכונה יש "תוכנה" הבנויה מטבלת מצבים בו כל שורה מהווה מצב, הפעילות תסתיים במצב "עצור".

אך כפי שראינו, מדובר במחשב דיגיטלי..  
מה לגבי העולם הקוונטי?  
אחד ההבדלים המהותיים בין העולם הקוונטי לעולם הקלאסי המוכר לנו מחיי היום-יום הוא שבעולם הקוונטי עצמים יכולים להיות במצב של '**סופרפוזיציה**'.   
סופרפוזיציה נובעת מהעובדה שברמה הקוונטית, חלקיקי החומר מתנהגים כגלים.  
ניתן להרכיב שני גלים, המתארים את החלקיק בשני מצבים שונים, זה על זה. כך מתקבל גל אחד שמבחינה מסוימת הוא גם בשני המצבים בו זמנית, אך בעצם אינו באף אחת מהם  
 (היות והוא שונה מכל אחד מהגלים בנפרד).  
איך מחשב קוונטי יתמודד עם הבעיה שלנו?

# מסקנות:

קשה לפספס את הקפיצה האדירה מ BB(4)ל BB(5)

מה שמוביל למסקנה המתבקשת:  
גם ∑(n)וגם S(n)הן לא פונקציות ניתנות לחישוב, הן פשוט גדלות מהר מדיי.

המסקנות האלה מעוררות עניין רב יותר בנושא אותו אנו חוקרים ע"פ המאמרים של פרופ' סקוט אהרונסון ועוד.

אנו שמים לב למרכזיותה של הבעיה, הקושי בהתקדמות בה, והאור שהיא יכולה לשפוך במקרה כזה על בעיות קריטיות אחרות בעולם המתמטיקה.

# סיכום:

בחנו לעומק את הבעיה המרתקת של הבונה העסוק,

הבנו למה היא כ"כ לא טריוויאלית ואת הקשר הישיר שלה לבעיית העצירה.

נסיים בתקווה שנוכל לחשוב על בעיות נוספות שנגזרות מבעיה זו,

כמו כן לחשוב על כיווני פיתרון אופציונליים, בלתי "כריעים" ככל שיהיו..

# ביבליוגרפיה:

1. <https://www.scottaaronson.com/blog/?p=2725>

2. <https://catonmat.net/busy-beaver>

3. <http://yeda.cs.technion.ac.il:8088/corpus/software/corpora/HeWiki_2013/html/10000-19999/109827.html>

4. <https://mathworld.wolfram.com/BusyBeaver.html?fbclid=IwAR2XryatVyTOh36ETgb-Z71KJlAPh8UwA33Mjg8GsDkaOT0ZSPOM7dD0qQ8>

5. <https://cseweb.ucsd.edu/classes/sp13/cse200-a/handouts/lec-02-0403.pdf>

6. <https://jeremykun.com/tag/busy-beaver/>

7. <https://arxiv.org/pdf/1311.1029.pdf>

8. <https://www.quantamagazine.org/the-busy-beaver-game-illuminates-the-fundamental-limits-of-math-20201210/>

9. <https://www.youtube.com/watch?v=CE8UhcyJS0I> - Busy Beaver Turing Machines –

10. <https://googology.wikia.org/wiki/Forum:Sigma_project>

11. <https://en.wikipedia.org/wiki/Busy_beaver>

12. <https://homepages.hass.rpi.edu/heuveb/Teaching/Logic/CompLogic/Web/Presentations/The%20Busy%20Beaver%20Problem.pdf>

13. <http://grail.cba.csuohio.edu/~somos/bb.html>

14. <https://www.semanticscholar.org/paper/Attacking-the-Busy-Beaver-5-Marxen-Buntrock/2dd93aba63fd464e0b28efd42a3208bf7fb8c785>

15. <https://math.stackexchange.com/questions/2738279/busy-beaver-function-growth-rate-compared-to-computable-functions>