מודלים חישוביים, חישוביות וסיבוכיות (חישוביות) - 67521

26 ביוני 2012

מרצה: גיא קינדלר
מתרגל: שאול אלמגור

"...one TM to rule them all..."

באדיבות בן מאירי

איני לוקחת אחריות על מה שכתוב כאן, so tread lightly
אין המרצה או המתרגל קשורים לסיכום זה בשום דרך.
הערות יתקבלו בברכה - noga.rotman@gmail.com. אהבתם? יש עוד! http://bit.ly/integrali

תוכן עניינים

Overview 0.1

קורס זה במובן מסויים הוא קורס במתמטיקה יותר מבמדעי המחשב.
מה אנחנו מנסים לעשות בקורס?
במבוא למדעי המחשב, $dast$ ואלגוריתמים ראינו אלגוריתמים לחישובים שונים. בקורס זה נראה מהו חישוב בצורה יותר פורמלית. חישוב הוא תהליך שפותר בעיה חישוביות. נראה כמה דוגמאות לבעיות חישוביות:

1. נתון גרף G. האם יש בו מסלול אוילר[1]?
2. בהנתן קבוצה מספרים $A = \{a_1, a_2, ..., a_n\}$ (נחשוב עליהם בתור מספרים מאוד גדולים), ומספר נוסף t, האם ניתן להציג את t כסכום של תת קבוצה של A?
3. בהנתן n זוגות של מילים מעל הא"ב $\{a, b\}$: $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, ..., \begin{pmatrix} x_n \\ y_n \end{pmatrix}$, האם יש סדרה סופית $a_1, a_2, ..., a_k$ (של מספרים) כך שהשרשור $x_{a_1}, x_{a_2}, ..., x_{a_k}$ (כאשר $a_1, .., a_k$ משמש כאן כאינדקס) שווה לשרשור:

$$y_{a_1}, y_{a_2}, ..., y_{a_k}$$

נשים לב שזה לא בהכרח אומר ש־$y_{a_1} = x_{a_1}$, שכן במקרה כזה פתרנו את הבעיה ־ פשוט ניקח את הסדרה $a_1 = 1$, וסיימנו. המטרה כאן היא למצוא שרשורים שווים זה לזה (בעיית הכרעה ־ האם קיים כזה). למשל:

$$x_1 = aba,\ y_1 = ab$$
$$x_2 = b,\ y_2 = ab$$
$$\Rightarrow a_1 = 1,\ a_2 = 2 \Rightarrow x_1x_2 = abab = y_1y_2,\ x_1 \neq y_1,\ x_2 \neq y_2$$

כעת נדבר על חישובים:

1. אם נרצה חישוב שעונה על שאלה 1 ־ האם קיים חישוב כזה? אם כן, האם קיים חישוב יעיל כזה?
כן ־ קיים משפט: בגרף יש מסלול אוילר אמ"מ מספר הצמתים שדרגתם אי זוגית הוא לא יותר מ־2.
אזי, כדי למצוא חישוב מתאים לשאלה 1, מספיק לבדוק מהו מספר הצמתים שדרגתם אי זוגית. קיים חישוב יעיל הלינארי (?!) בגודל הקלט.
2. לגבי שאלה 2 ־ קיים, למשל, אלגוריתם דינמי: לחשב את קבוצת כל המספרים שניתן להגיע אליהם כסכום של a_1, a_2, לאחר מכן לקחת את המספרים האלו ולהוסיף או לא להוסיף את a_3, ולהוסיף את התוצאות לקבוצת המספרים המקורית, וכך הלאה.
במקום תכנון דינמי, אפשר לבדוק את כל הסכומים:

$$\sum_{i \in S} a_i,\ \forall S \in \{1, ..., n\}$$

יש 2^n סכומים כאלו (כי יש 2^n תת קבוצות), ולכן בעוד שהחישוב עצמו מאוד פשוט, הוא ארוך מאוד (אקספוננציאלי) ולא יעיל במיוחד. לא ידוע לנו על חישוב יעיל לבעיה זו.
3. לגבי שאלה 3, אפשר לפעול בצורה מאוד נאיבית ־ לנסות את כל הסידורים האפשריים (כאן יש חשיבות לסדר). כאן יכול להיות שאי אפשר לסדר תת קבוצה של הזוגות האלו, אולם אפשר אם חלק מהזוגות חוזרים בסדרה. לכן, אין גבול אורך הסדרה, ולכן יש אינסוף אפשרויות, לכן לא ניתן לעבור על כל האפשרויות. היינו רוצים אם כך אלגוריתם אחר, אבל מתברר שאין כזה ־ כלומר, אין חישוב שפותר את הבעיה הזו. נוכיח זאת במסגרת הקורס.

ראינו אם כך שיש בעיות חישוביות שהן קלות, יש בעיות חישוביות שהן קשות, וישנן בעיות חישוביות שאין אף אלגוריתם שפותר אותן.
יש גם בעיה אחרת שהרבה מאיתנו נתקלים בה:
בהנתן תוכנית מחשב M בשפת C, האם היא נכנסת ללולאה אינסופית (כשמריצים אותה בלי שום קלט, אפשר גם לדבר על קלט ספציפי, או האם קיים קלט שמכניס אותה ללופ, וכו'), כלומר האם היא תסיים או לא תסיים?
<u>אין</u> שום קומפיילר שנכניס לתוכו את התוכנית ויוכל לומר לנו שהתוכנה תסיים\לא תסיים. בעיה זו היא גם באותו מעמד כמו בעיה 3 מעלה. גם את זה נראה במסגרת הקורס. זה בעצם יותר עמוק ממדעי המחשב בלבד ־ אפשר לשאול, למשל, בהנתן משפט מתמטי, האם קיימת הוכחה שמסתמכת על קבוצת אקסיומות מסויימת? גם שאלה זו לא ניתנת לפיתרון בעזרת מחשב.

שלושת חלקי הקורס יהיו:

[1] תזכורת: מסלול אוילר הוא מסלול שמתחיל באחד הקודקודים והולך על כל קשתות הגרף פעם אחד, ולא יותר מפעם אחת.
יש כמובן גרפים שאין בהם מסלול אוילר.

1. אוטומטיים סופיים: כדי להראות שלא קיים אלגוריתם, צריך להגדיר את המושג "אלגוריתם". מסתבר שההגדרה הזו לא פשוטה. נלמד את המודל של אוטומטיים סופיים, שקל יותר להשתמש בו מאשר להגדיר מההתחלה את המושג אלגוריתם.
2. חישוביות: אילו בעיות הכרעה ניתן להכריע בעזרת מחשב? ניתן כאן תשובה חלקית אך מספקת.
3. סיבוכיות: מה ניתן לפתור בצורה יעילה בעזרת מחשב? לדוגמא, את שאלה מס' 2 אנו לא יודעים אם אפשר לפתור אותה בצורה יעילה (בזמן פולינומיאלי). באופן כללי, "יעילה" יכול להתייחס לזמן, לזיכרון, וגורמים נוספים. כלומר, אילו משאבים נדרשים כדי לפתור שאלות. לגבי שאלה 2, נראה את ההשערה המפורסמת $NP \stackrel{?}{=} P$, וכן אם $NP \neq P$, אז לשאלה 2 אלגוריתם יעיל (במקרה זה - פולינומי).

0.2 מנהלות

0.2.1 מקורות

חשוב להגיע לשיעורים, אי אפשר ללמוד רק מהספרים.

- ספר הקורס הוא (למרות שלא מתחייב שכל דבר שילמד יהיה על פיו, אולי הכל לא יהיה מכוסה בו):

  Sipser: Introduction to theory of computing

  ממליץ לא ללמוד מהמהדורה העברית, אלא מהמקורית. הוא מקור טוב מאוד כמעט לכל החומר.
- חוץ מממנו יש המון סיכומים משנים קודמות (מופיע בדפי הקורס של שנים קודמות - של איילת מזרחי מצויין).
- סיכומים מהשנה הנוכחית.
- הקלטות השיעור משנה שעברה.

0.2.2 הרכב הציון

כרגע יש שביתה של הסגל הזוטר. במידה והשביתה לא תגמר עד שבוע הבא, נתמודד עם זה אז.

- תרגילים:
  - 20%, לא מגן, 2 הגרועים לא יחשבו (מתוך בערך תרגיל תוך שבוע, בהתחשב במצב השביתה).
  - השנה אין ראיונות יש "ראיונות כתובים", כלומר בחנים: כל אחד יכול לבחור מתי הוא מגיע מתוך כמה אפשרויות, ונבחן על כמה שאלות מתוך שיעורי הבית. זה "רשות" - כלומר אם לא מעוניינים לגשת ל"ראיונות הכתובים", אפשר לקבל חומר רשות מהמרצה במקום לעשות את הראיונות הכתובים, ולהבחן אליו בשאלה נוספת בבחינה[2] (נניח כרגע שיהיה עוד זמן במבחן לשאלה הזו)[3]. בבחנים יופיעו השאלות כפי שהופיעו בתרגיל עצמו (לא וריאציות).
  - שני "הגרועים" הם הדרך לעקוף את בקשות הפטור למיניהן (מלבד מילואים שלא יחשבו שם).
  - 20% לא עוזרים למי שלא עובר את המבחן.
- מבחן:
  - 80%.

0.2.3 פניות, בקשות, טענות

לרב, המתרגלים הם הכתובת הראשונה.
לטענות כלליות (לא אישיות) אפשר לפנות לנציגות האגודה.[4]

[2] כאן המרצה השתמש במונח "הפתרון הסופי". נשמע מעודד.
[3] "למה אמרתי את זה? אני לא בדיוק זוכר".... "אם הציונים יהיו נורא טובים, גם זה לא ממש יעזור לכם".
[4] "אין לי אינטרס שאף אחד יכשל". נחזור לציטוט הזה בסוף הקורס.

חלק I
שיעורים

1 שפות

1.1 אוטומטים - הגדרות בסיסיות

אוטומט הוא מודל חישובי מוגבל - הוא לא יכול לעשות את כל מה שהמחשב יכול לעשות, אך הוא יכול לפתור בעיות הכרעה מסויימות.

נסמן את הא"ב שלנו ב-$\Sigma = \{a, b\}$.

קלט עבור האוטומט יהיה מילה מעל הא"ב הזה - כלומר, שרשור של אותיות. נסמן ב-Σ^* את אוסף המילים מעל הא"ב Σ. למשל, $abba \in \Sigma^*$.

הבעיה שהאוטומט יפתור: האם מילה מ-Σ^* מקיימת איזשהיא תכונה?

דוגמא לבעיה חישובית שהאוטומט יכול לפתור - האם מילה נתונה מכילה רצף באורך $3 \leq$ של אחת האותיות? למשל, המילה $abba$ אינה מכילה רצף באורך 3, ולכן האוטומט אמור לדחות המילה הזו. לעומת זאת, המילה $baaa$ תתקבל ע"י האוטומט כבעלת התכונה הזו.

הבה נבנה אוטומט שימצא את התכונה הזו:

האוטומט פועל בגרף: *להכניס גרף*

זהו אוטומט מאוד פשוט לבעיה חישובית מאוד פשוטה, ואם זאת, להוכיח זאת בצורה פורמלית זה לא דבר פשוט בכלל.

התיאור של האוטומט בו דנו בשיעור הקודם סופי. בכך זה דבר שמאוד דומה למה שאנחנו חושבים על חישוב - משהו שמכונה עושה בצורה סופית. כמו כן, המכונה שלנו פועלת בצורה מקומית (במקום תיאור של פונקציה למשל), ולכן הדבר הזה דומה למה שאנחנו חושבים באינטואיציה על חישוב. זהו מודל שהוא חלש ממה שהמחשב באמת יכול לעשות, אבל הוא שימושי.

היום נרצה להגיד את המודל הזה מההתחלה עד הסוף - תיאור האוטומט, ריצת האוטומט, ומה הוא עושה.

הגדרה 1.1 א"ב הוא Σ קבוצה לא ריקה, סופית. לאיברי Σ קוראים אותיות\תווים.

הגדרה 1.2 מילה[5] איבר של Σ^n עבור $n \in \mathbb{N}$ (כל מילה היא באורך סופי), או ε - המילה הריקה (מילה באורך אפס).

הערה 1.3 נניח ש-ε לא בא"ב, כלומר סימן זה יהיה שמור תמיד למילה הריקה.

למשל:

$$aba \in \Sigma^3,\ abba \in \Sigma^4, ...$$

סדרה אינסופית של אותיות לא נקרא מילה.

הגדרה 1.4 Σ^* הוא אוסף כל המילים מעל הא"ב Σ.

למשל, $\varepsilon \in \Sigma^*$, $aba \in \Sigma^*$.

הגדרה 1.5 כל תת קבוצה $L \subseteq \Sigma^*$ נקראת שפה מעל Σ.

לדוגמא, אוסף המילים מעל $\Sigma = \{a, b\}$ המכילות רצף של לפחות 3 אותיות זהות הוא שפה. האוטומט מהשיעור הקודם בדיוק מזהה את השפה הזו. בהמשך נראה את הקשר בין אוטומטים לשפות (האם אוטומט יכול להכריע שפה או לא). השפה הזו היא אינסופית, אך יכולה גם להיות סופית - יכולה להכיל למשל מילה אחת, שתיים או אפס מילים.

עוד דוגמאות לשפות בא"ב $\Sigma = \{a, b\}$:

- השפות הטריוויאליות:
  - $\Sigma^* \supseteq L = \phi$ - השפה הריקה.
  - $\Sigma^* \supseteq L = \Sigma^*$.
- $L = \{aba\}$.
- $L = \{\varepsilon\}$[6].

[5] בשונה ממה שאנחנו מכירים על שפה, כאן לא יהיו לנו "משפטים".

[6] נשים לב ששפה זו היא אינה השפה הריקה!

- שפת המילים שבהן מספר המופעים של "a" הוא:
  - מתחלק ב-3 עם שארית 1[7].
  - ראשוני[8].

מעל א"ב אחר - כלומר שאינו $\{a,b\}$:

- שפת כל המשפטים המתמטיים הניתנים להוכחה מהאקסיומות[9].

1.1.1 אוטומטים סופיים דטרמיניסטיים - DFA

הגדרה 1.6 אוטומטון סופי דטרמיניסטי $\text{DFA}-(\text{Deterministic Finite Automaton})$ הוא חמישיה סדורה (Q,Σ,δ,q_0,F), כאשר:

- Q היא קבוצה סופית של מצבים.
- $q_0\in Q$ הוא האיבר ההתחלתי[10].
- $F\subseteq Q$ היא קבוצת המצבים המקבלים[11] - בדוגמא מהשיעור הקודם - קבוצה זו מכילה רק מצב אחד - הסופי.
- δ היא פונקציה הנקראת פונקציית המעברים, המקיימת:

$$\delta : Q\times\Sigma\to Q$$

כלומר, המקור שלה - מכפלה קרטזית בין קבוצת המצבים והא"ב (האות שכרגע האוטומט מקבל), והטווח שלה - קבוצת המצבים, כאשר המצב שיתקבל הוא זה שהאוטומט יעבור אליו אם במצב הנוכחי הוא מקבל אות מסויימת. בדוגמא מהשיעור הקודם, הפונקציה הזו מיוצגת ע"י החצים בציור.
במילים אחרות, אם האוטומט נמצא במצב q וקורא את האות α, אז המצב הבא של האוטומט יהיה $\delta(q,\alpha)$.

הגדרה 1.7 יהא $A=(Q,\Sigma,\delta,q_0,F)$ אוטומט ד' ס'.
נגדיר את פונקצית ההרחבה של δ - δ^* המקיימת:

$$\delta^* : Q\times\Sigma^*\to Q$$

באינדוקציה על מילים באורך $\mathbb{N}\ni n=0,1,2,...$[12]:

- עבור מילים באורך 0 ומצב $q\in Q$:

$$\delta^*(q,\varepsilon)=q$$

- עבור מילים באורך 1, כלומר $a\in\Sigma$, ומצב $q\in Q$:

$$\delta^*(q,a)=\delta(q,a)$$

- נניח שהגדרנו את δ^* עבור מילה באורך n. עבור מילה w באורך $n+1$ ומצב q (עוברים על מילה "כמנהג הגויים" משמאל לימין), נכתוב $w=za$, כאשר $|z|=n$, $a\in\Sigma$, ונגדיר:

$$\delta^*(q,w)=\delta^*(q,za)=\delta(\delta^*(q,z),a)$$

מהו אוסף המילים שהאוטומט מקבל?

[7] שפה זו כן אפשר להכריע בעזרת אוטומט סופי.
[8] שפה זו אי אפשר להכריע בעזרת אוטומט סופי - נראה בהמשך.
[9] שפה זו אי אפשר להכריע בעזרת אוטומט סופי - נראה זו בהמשך.
[10] יש אוטומטים שיש בהם רק איבר אחד.
[11] יכולה להיות גם ריקה - ממנה נקבל את השפה הריקה. זה לא נראה מעניין במיוחד, אבל אולי נרצה לשנות את האוטומט הזה בהמשך. יש לכך שימושים.
[12] כלומר $\delta^*(q,w)$ זה מצב שאליו נגיע אם נתחיל במצב q ונעבור על המילה w.

הגדרה 1.8 נגיד שמילה $w \in \Sigma^*$ מתקבלת ע"י האוטומט A, אם $\delta^*(q_0, w) \in F$.

הגדרה 1.9 השפה המתקבלת ע"י האוטומט A תסומן ע"י $L(A)$, ומוגדרת ע"י:

$$L(A) = \{ w \in \Sigma^* | \, \delta^*(q_0, w) \in F \}$$

דוגמא:
L מעל $\Sigma = \{a, b\}$ היא אוסף המילים שבהם $\#a = 1 \, (mod 3)$.
איך נבנה אוטומט לשפה הזו?
מצב ההתחלתי - q_0, "כי כך זה בכל אוטומט".
המצב של האוטומט ייצג איכשהוא את מספר ה-a שהיו במילה עד עכשיו:
ציור של האוטומט
q_1 הוא המצב אליו יגיעו כל המילים שמספר ה-a בהם מחלק את 3 בשארית 1, ולכן הוא יהיה המצב המקבל, והוא בלבד.
נרצה לתת ייצוג פורמלי של האוטומט הנ"ל לפי ההגדרה שלנו:
$(Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$.
- $\Sigma = \{a, b\}$.
- q_0 (הוא פשוט עצמו).
- $F = \{q_1\}$.
- נגדיר את $\delta : Q \times \Sigma \to Q$. נייצג את הפונקציה בעזרת טבלה:

	a	b
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_0	q_2

השלב הבא הינו להוכיח שהשפה של האוטומט שהגדרנו היא בדיוק השפה שרצינו לייצג. כיצד עושים על זה? בשיעור הבא!

הגדרה 1.10 REG הוא אוסף כל השפות L שקיים A DFA המקבל אותן:

$$REG = \{L | \exists A, \; A \; is \; DFA, \; L(A) = L\}$$

הסיבה לכך שהאוסף נקרא REG (רגולרית) תתגלה לנו בהמשך.
יהא A האוטומט:

$$A = (Q, \Sigma, \delta, q_0, F)$$

כאשר:

$$Q = \{q_0, q_1, q_2\}, \; \Sigma = \{a, b\}, F = \{q_1\}$$

ו-δ נתון ע"י הטבלה:

	a	b
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_0	q_2

טענה 1.11

$$L(A) = \{w \in \Sigma * | \#_a(w) = 1 \, (mod 3)\}$$

הוכחה: נוכיח באינדוקציה. בהוכחות כאלו האינדוקציה עצמה תמיד קלה, החלק הקשה הוא למצוא את טענת האינדוקציה המתאימה:
החוכמה היא לדבר על איזה מצב מגיעה כל מילה, ולא רק לדבר על המילים המגיעות למצב מקבל.
נוכיח באינדוקציה על אורך המילה w - נסמנו ב-n, את הטענה הבאה:

$$\delta^*(q_0, w) = q_{\#_a(w)(mod3)}$$

בסיס האינדוקציה: $n = 0$:

$$\delta^*(q_0, \varepsilon) \overset{def}{=} q_0$$

ואכן $\#_a(\varepsilon) = 0$.
נניח נכונות הטענה עבור n, ונראה את נכונות הטענה עבור $n+1$.
תהא $w\tau$[13] מילה באורך $n+1$. עפ"י הנחת האינדוקציה:

$$\delta^*(q_0, w) = q_{\#_a(w)(mod3)}$$

אם $\tau = a$: אז $\#_a(w\tau)(mod3) = (\#_a(w)+1)(mod3)$. ואכן:

$$\delta^*(q_0, wt) = \delta^*(q_0, wa) = \delta(\delta^*(q_0, w), a)$$
$$\overset{ind.}{=} \delta\left(q_{\#_a(w)(mod3)}, a\right) \overset{\delta\text{'s def}}{=} q_{((\#_a(w)+1)(mod3))}$$

אם $\tau = b$: אז $\#_a(w\tau)(mod3) = \#_a(w)(mod3)$, ואמנם:

$$\delta*(q_0, w\tau) \overset{\text{like before}}{=} \delta\left(q_{\#_a(w)(mod3)}, b\right) \overset{\delta\text{'s def}}{=} q_{\#_a(w)(mod3)}$$

כנדרש. ■

לא לכל שפה יש אוטומט.

1.1.2 תכונות הסגירות של REG

יהיו L_1, L_2 שפות רגולריות מעל א"ב Σ. האם:

- $L_1 \cup L_2$ בהכרח רגולרית?
- $REG \ni L_1 \cap L_2$ בהכרח?
- $REG \ni \Sigma* \setminus L_1$ בהכרח? (בתרגיל הבית).

פעולת השרשור: יהיו $w, x \in \Sigma*$, $w = w_1w_2...w_n,\ x = x_1x_2...x_m$. אזי השרשור של שתי המילים מוגדר להיות:

$$wx = w_1...w_nx_1...x_m$$

וכן $\varepsilon x = x,\ \varepsilon\varepsilon = \varepsilon$.
עבור $L_1, L_2 \subseteq \Sigma*$, פעולת השרשור של שתי השפות תוגדר להיות:

$$L_1 \circ L_2 = \{wx | w \in L_1, x \in L_2\}$$

- האם בהכרח $L_1 \circ L_2 \in REG$?

על השאלות האלו התשובה היא כן. כדי להוכיח את זה צריך להרכיב מהאוטומטים הנתונים אוטומט חדש. נראה אלגוריתם לזה.

14/3/2012

טענה 1.12 יהיו L, M שפות רגולריות מעל א"ב Σ. אזי $REG \ni L \cap M$.

[13] "שתי חברות נכבדות באב היווני, שצריכות עכשיו את כל התמיכה שאפשר לתת להן בגלל המצב הכלכלי"

הערה 1.13 זוהי למעשה שאלה אלגוריתמית. תרגום: בהנתן שני אוטומטים (אחד של השפה M ושני של השפה L), מצא אוטומט של השפה $L \cap M$.

הוכחה: הרעיון - לעשות "אוטומט מכפלה" - להריץ את שני האוטומטים הנתונים במקביל. כל מצב באוטומט חדש יתאים לזוג מצבים משני האוטומטים המקוריים.
יהיו $A = (Q, \Sigma, \delta, q_0, F)$ ו-$B = (P, \Sigma, \lambda, p_0, G)$ אוטומטים (DFA) המזהים את השפות L, M בהתאמה ($A,\ M \Rightarrow B \Rightarrow L$). נבנה מהם אוטומט C כך ש-$L(C) = L \cap M$:

- מצבי C: $Q \times P$ (כל איבר בקבוצת המצבים הוא זוג סדור של מצב מאוטומט A ומצב מאוטומט B).
- א"ב: Σ.
- מצב התחלתי: (q_0, p_0).
- קבוצת המצבים המקבלים: $F \times G$.
- פונקציית המעברים:

$$\varphi : (Q \times P) \times \Sigma \to (Q \times P)$$

היא תוגדר ע"י השיוויון הבא:

$$\varphi((q,p),\alpha) = (\delta(q,\alpha), \lambda(p,\alpha))$$

נותר להוכיח שהשפה המתקבלת מהאוטומט החדש הינה השפה המבוקשת, כלומר כי $L(C) = L \cap M$. כבר ראינו איך עושים את זה בשיעור הקודם - מבינים לאן כל מילה מגיעה באוטומט (כל מילה, לא רק המילים המגיעות למצב מקבל - זה החלק הקשה), ואז להוכיח באינדוקציה.

טענה 1.14 לכל מילה $w \in \Sigma*$, מתקיים:

$$\varphi^*((q_0,p_0),w) = (\delta^*(q_0,w), \lambda^*(p_0,w))$$

הוכחה: באינדוקציה על $|w| = n$:
בסיס: $|w| = 0$, כלומר w היא המילה הריקה. אזי:

$$\varphi^*((q_0,p_0),\varepsilon) \overset{def}{=} (q_0,p_0) \overset{def}{=} (\delta^*(q_0,\varepsilon), \lambda^*(p_0,\varepsilon))$$

נניח נכונות הטענה עבור מילים עד אורך n[14], נוכיח עבור $n+1$: נביט במילה $w = u\alpha$, עבור $|u| = n$. אזי:

$$\varphi^*\left((q_0,p_0), \overbrace{u\alpha}^{w}\right) = \varphi(\varphi^*((q_0,p_0),u),\alpha) \overset{induction}{=}$$
$$\varphi((\delta^*(q_0,u), \lambda^*(p_0,u)),\alpha) \overset{def\ \varphi}{=} (\delta(\delta^*(q_0,u),\alpha), \lambda(\lambda^*(p_0,u),\alpha))$$
$$\overset{def\ \delta^*,\lambda^*}{=} (\delta^*(q_0,w), \lambda^*(p_0,w))$$

וכאן מסתיימת הוכחת האינדוקציה.[15] ■

לסיום, במקום להוכיח את שני צידי הטענה, נוכיח את השניים יחדיו!
$(\delta^*(q_0,w), \lambda^*(p_0,w)) \in \Leftrightarrow (\delta^*(q_0,w) \in F) \wedge (\lambda^*(p_0,w) \in G) \Leftrightarrow w \in L(A) \wedge W \in L(B) \Leftrightarrow w \in L \cap M$
מטענת העזר $w \in L(C) \Leftrightarrow \varphi^*((q_0,p_0),w) \in F \times G \Leftrightarrow F \times G$. ■

טענה 1.15 $L, M \in REG$. אז $L \cup M \in REG$.

[14] אינדוקציה שלמה
[15] "ובטוקבקים היו שואלים: "למה לא דיברת על המצב בסוריה?"".

הוכחה: יהיו $A=(Q,\Sigma,\delta,q_0,F)$ ו־$B=(P,\Sigma,\lambda,p_0,G)$ אוטומטים (DFA) המזהים את השפות L,M בהתאמה ($L\Rightarrow A,\ M\Rightarrow B$). נבנה אוטומט C כמו קודם, למעט קבוצת המצבים המקבלים, שתהיה:

$$(F\times P\cup Q\times G)$$

אזי, יש להראות כי $w\in L\cup M$ אמ"מ $\varphi*((q_0,p_0),w)\in(F\times P\cup Q\times G)$. ההכוחה עצמה נשארת כתרגיל לקורא החרוץ (נשים לב כי טענת העזר מההוכחה הקודמת עדיין נכונה, גם עבור האוטומט הזה, שכן היא לא התייחסה כלל לקבוצת המצבים המקבלים). ■

באיזשהוא שלב נבין איזו תכונה הופכת שפה לרגולרית, ואז נראה את הטענות הנ"ל בעזרת הקריטריון שנמצא. הסיבה שאנחנו עושים את העבודה הזו בכל זאת, היא כדי לקבל ניסיון בכל מיני דברים: הוכחות פורמליות, ובכל מיני שיטות של בניית מודלים חישוביים או "מכונות" חישוב. אחד העקרונות החשובים הוא זה שעשינו עכשיו - עקרון המכפלה.

עד עכשיו הראינו את הסגירות של REG לאיחוד וחיתוך.
הדבר הבא הטבעי לדבר עליו הוא השלמה - $\Sigma*\backslash L$ הוא מקרה קל והוא בתרגיל הבית.
המקרה הבא פחות טריוויאלי - $L\circ M$ שרשור של שתי שפות רגולריות:
שאלה: $L,M\in REG$. האם בהכרח

$$L\circ M=\{uw|u\in L,w\in M\}$$

רגולרית?
נביט בשתי שפות פשוטות בא"ב $\Sigma=\{a\}$, $L=\{a^n|n=2\,(mod7)\}\ ,\ M=\{a^n|n=1\,(mod31)\}$. איך יראה אוטומט עבור השפה L?
ציור של אוטומט
יהיו לו מצב התחלתי q_0, האות היחידה שמתקבלת היא a, המצב q_2 תהיה המצב המקבל היחידי.
לגבי M, q_0 שוב מצב התחלתי, a מוביל ל־q_1 מצב מקבל, דומה מאוד לקודם, רק כאן אורך המעגל הוא 31 מצבים.
איך אפשר להרכיב משני האוטומטים הללו אוטומט שמקבל את השרשור?
הצעה אחת היתה: "הולכים" עד מצב מקבל של האוטומט הראשון (בפעם הראשונה!!), ומשם "קופצים" למצב ההתחלתי של המצב השני, עד שמגיעים למצב מקבל של האוטומט השני. אבל זה לא עובד, למשל שרשור של $aa\in L$, ולאחריה a^{32}: המילה המתקבלת היא $aaa^{32}=a^{34}$. זה בסדר. אבל את השרשור $a^9a^{32}=a^{41}$ האוטומט הזה ידחה, למרות ש־$a^{41}\in L\circ M$.
הצעת פתרון: לבנות אוטומט עם $7\cdot31$ מצבים, ואז בכל צעד נוכל לדעת מה השארית של 7 ושל 31, ולכן פתרון זה יעבוד. אבל יעבוד ספציפית לדוגמא הזו של שרשור, ולאו דווקא לדוגמאות אחרות. נרצה למצוא פתרון כללי.
נראה פתרון "מפוקפק", בעזרת הוספת הגדרה. זה לא ממש יענה לנו אם $L\circ M\in REG$, אבל לאחר מכן נעשה שלב נוסף - נראה שאם יש אוטומט לא דטרמיניסטי המזהה שפה כלשהיא, אפשר לבנות אוטומט דטרמיניסטי שיזהה את השפה הזו, ואז נסיים את מה שאנו רוצים להראות.

1.1.3 אוטומטים סופיים לא דטרמיניסטיים - NFA

נוסיף חץ לאוטומט שיצרנו מ־q_1 ב־A ל־q_0 ב־B - לא סביר לממש אותה, שכן יש למכונה זו שתי אפשרויות מ־q_1 לעבור ל־q_2. לאוטומט לא דטרמיניסטי יש כמה דרכים לרוץ, אולם לעיתים לא יהיו לו אף דרכים לרוץ.

הגדרה 1.16 [16]אוטומט לא דטרמיניסטי - NFA, הוא חמישיה $A=(Q,\Sigma,\delta,Q_0,F)$, כאשר:

- Q היא קבוצת מצבים סופית.
- Σ הוא א"ב.
- $Q_0\subseteq Q$ היא קבוצת המצבים ההתחלתיים.
- $F\subseteq Q$ הוא אוסף המצבים המקבלים.
- δ היא פונקציית המעברים:

$$\delta:Q\times\Sigma\to\mathcal{P}(Q)$$

כאשר $\mathcal{P}(Q)\subseteq Q$ היא קבוצה של מצבים.

הערה 1.17 יכול להיות ש־δ תיתן קבוצה ריקה - במצב כזה המילה לא יכולה לרוץ על האוטומט עד הסוף.

[16]שימו ♡ לעדכון בהגדרה בתאריך 19/03 שהוסף להגדרה זו ולהגדרה הבאה.

הגדרה 1.18 בהנתן NFA כנ"ל, נגדיר את פונקציית המעברים המורחבת δ^*:

$$\delta^* : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$$

נרצה כי בהנתן $\delta^*(S,w)$, מה שיתקבל הוא כל המצבים שנוכל להגיע אליהם במידה והתחלנו ב-S - יכול להיות ו"נתקע" ולא נגיע לסוף. כדי לעמוד בדרישה זו, עבור $S \subseteq Q$ ו-$w \in \Sigma^*$, נגדיר באינדוקציה על $|w|$ את $\delta^*(S,w)$ באופן הבא:

- עבור $|w| = 0$:

$$\delta^*(S,\varepsilon) = S$$

- נניח שהגדרנו את δ^* עבור מילים באורך n. עבור $|w| = n+1$, נסמן $w = u\alpha$, כאשר $\alpha \in \Sigma$, $|u| = n$:

$$\delta^*(S,u\alpha) = \bigcup_{q \in \delta^*(S,u)} \delta(q,\alpha)$$

נשים כאן לב כי הסיבה לשינוי כאן מההגדרה של DFA הוא מכך ש-δ מוגדרת על מצב יחיד, ולא על קבוצת מצבים, ועל כן יש כאן צורך באיחוד ע"מ לבטא את הביטוי $\delta(\delta^*(S,u),\alpha)$.

תהי L מעל $\sum = \{a,b\}$:
מילים שאורכן מתחלק ב-5 עם שארית 1:
ציור של אוטומט
בעזרת המצבים נספור את מספר האותיות במילה, כאשר המצב q_1 הוא המצב המקבל, מכל מצב יוצאים שני חצים, אחד לכל אות בא"ב.
נוסיף קריטריון:
שלוש האותיות האחרונות בהן שוות:
ציור של אוטומט משופצר
בתור התחלה נשכח מהמצב המקבל הקודם. נוציא מ-q_3 שני מסלולים שונים שיביאו לשני המצבים מקבלים של האוטומט החדש, אחד לכל אות בא"ב, כאשר בשני המסלולים נמשיך קדימה רק אם האות החדשה שווה לאות הקודמת: $q_5 - q_7$ ל-a, $q_8 - q_{10}$ ל-b. עתה קיבלנו אוטומט שאינו דטרמיניסטי, שכן מהמצב q_3 אפשר להמשיך לאחד משני מסלולים - להמשיך ל-q_4, או להמשיך ל-q_5 או q_8, בהתאם לאות שהתקבלה. הקביעה אם מילה בשפה תנבע מהתשובה לשאלה, <u>האם קיימת ריצה כלשהיא</u> בה המילה תגיע למצב מקבל?
נביט למשל במילה a^{21} - אורכה מתחלק ב-5 עם שארית 1, וכן היא בשפה, שכן <u>יש</u> ריצה של האוטומט בה היא מגיעה למצב מקבל.
זו איננה מכונת חישוב, אפשר לחשוב עליה ככלי שיעזור למישהו אחר להחליט\להוכיח אם המילה בשפה.
נביט כעת במילה $b^{19}a^2$ - היא גם מילה באורך 21, אבל היא מילה שלא ניתן לקבלה ע"י האוטומט הזה.

הבה נשכלל את השפה L:
מילים שאורכן מתחלק ב-5 עם שארית 1 ושלוש האותיות האחרונות בהן שוות, או שמספר האותיות בהן מתחלק ב-4.
כיצד כעת נעדכן את האוטומט?
אוטומט עוד יותר מעודכן
לעשות את q_4 למצב מקבל לא יעבוד - לא יעבוד עבור למשל מילה באורך 8.
במקום זאת, ניקח אוטומט חדש בעל 4 מצבים - r_0, r_1, r_2, r_3, כאשר עוברים ממצב r_i למצב r_{i+1} ומ-r_3 ל-r_0 ע"י שני חצים, כל אחד לאות בא"ב, והמצב המקבל הוא r_0. אוטומט זה לכשעצמו הוא אוטומט סופי דטרמיניסטי.
נוסיף אותו לאוטומט הקודם בצורה הבאה: נכריז על קבוצת מצבי ההתחלה כ-$\{q_0, r_0\}$, כלומר, ניתן להתחיל בכל אחד משני המצבים הללו.
איך נוכיח שזה באמת אוטומט של השפה הזו?
אם המילה בקטגוריה הראשונה, ראינו שקיים לה מסלול מקבל באוטומט הראשון.
אם המילה בקטגוריה השניה, ראינו שקיים לה מסלול מקבל באוטומט השני.
מצד שני, יש להראות כי מילים שאינן באף אחד מהקטגוריות לא מתקבלות - אבל, ראינו שאם מילה לא בקטגוריה הראשונה, היא אינה מתקבלת באוטומט הראשון, וכן באוטומט הראשון מתקבלות רק מילים בקטגוריה הראשונה, ולכן לא יתקבלו בה מילים נוספות, כלומר מילים שאינן בקטגוריה השניה או בראשונה. עבור הקטגוריה השניה כנ"ל - אם מילה אינה בקטגוריה השניה, היא אינה מתקבלת באוטומט השני, וכן באוטומט זה מתקבלות רק מילים בקטגוריה השניה, ולכן לא יתקבלו בה מילים נוספות, כלומר מילים שאינן בקטגוריה הראשונה או השניה.
לכן, האוטומט שבנינו אכן מקבל את השפה L שהגדרנו, כנדרש.

נחזור לבעיה המקורית שלנו - נביט בשתי שפות רגולריות L, M. אזי לכל אחת מהן יש אוטומט המכריע אותן (DFA).
כיצד נמצא אוטומט לא דטרמיניסטי שמוצא את השרשור שלהן $L \circ M$?
נביט באוטומט A המקבל את L, והאוטומט B המקבל את M. אזי, נגדיר אוטומט חדש לא דטרמיניסטי בצורה הבאה:
לכל קשת שנכנסת למצב מקבל ב-A, נשכפל קשת שעוברת למצב ההתחלתי ב-B (זה DFA ולכן יש רק מצב התחלתי יחיד).

21/03/2012

טענה 1.19 יהיו $L, M \in REG$ מעל Σ. אזי קיים NFA שמזהה את השפה $L \circ M$.

הוכחה: יהיו A, B אוטומטים דטרמיניסטיים המזהים את השפות L, M בהתאמה. נבנה אוטומט NFA המזהה את $L \circ M$:

- הא"ב - Σ.
- קב' המצבים: האיחוד הזר בה"כ (אם לא היה זר מראש, אפשר היה לשנות את שמות המצבים כדי להפוך את האיחוד לזר) של קב' מצבי A וקבוצת מצבי B.
- קב' המצבים המקבלים: אוסף המצבים המקבלים של B.
- קב' מצבי ההתחלה: הקבוצה המכילה את מצב ההתחלה של A, וכמו כן, אם $\varepsilon \in L$ נוסיף גם את מצב ההתחלה של B.
- פונקציית המעברים: נגדיר את $\delta_C(q, \alpha)$ לכל מצב q של C ולכל $\alpha \in \Sigma$ באופן הבא:
  - אם q הוא מצב של B, נגדיר:

$$\delta_C(q, \alpha) = \{\delta_B(q, \alpha)\}$$

  - אם q הוא מצב של A, נגדיר את הקבוצה $\delta_C(q, \alpha)$ להכיל את המצב $\delta_A(q, \alpha)$. כמו כן, אם $\delta_A(q, \alpha) \in F_A$ (כלומר מצב מקבל באוטומט A), נוסיף ל-$\delta_C(q, \alpha)$ את המצב ההתחלתי של האוטומט B.

נשים לב כי בהגדרה זו, אם $\varepsilon \in M$, אז המצב ההתחלתי של B הוא מצב מקבל, ואז אין בעיה עם שרשורים מסוג $L_1 \circ \{\varepsilon\}$, שכן בסוף של מילה אחד המצבים האפשריים להמשך הוא המצב ההתחלתי של B, והוא כאמור מקבל.
השלב הבא הוא להוכיח כי האוטומט שבנינו באמת מזהה את שפת השרשור. עשינו כבר דברים כאלה באינדוקציה, ולכן לא נעשה את זה בכיתה (אולי בתרגיל). ■

משפט 1.20 $A = (Q_A, \Sigma, \delta_A, Q_0, F_A)$ הוא NFA. אזי, קיים DFA $B = (Q_B, \Sigma, \delta_B, q_0, F_B)$ כך ש-$L(A) = L(B)$.
כמו כן, $|Q_B| \leq 2^{|Q_A|}$.

הערה 1.21 החסם הזה הוא אפילו די הדוק - אולי נראה בתרגול דוגמא שאכן כדי לבנות NFA מתאים לשפה רגולרית B, מתקבל שיוויון.

הוכחה: הרעיון - מצבי B יהיו תת-קבוצות של מצבי A.
נגדיר את B כדלהלן:

- $Q_B = \mathcal{P}^{(Q_A)}$, דהיינו קבוצת המצבים של B היא תת קבוצות של מצבים מ-Q_A. אכן מספר המצבים שמתקבל הוא $2^{|Q_A|}$.
- $q_0 = Q_0$, דהיינו מצב ההתחלה ב-B הוא קבוצת המצבים ההתחלתיים ב-B.
- $F_B = \{S \subseteq Q_A | \, S \cap F_A \neq \phi\}$.
- $\delta_B(S, \alpha) = \delta_A^*(S, \alpha) = \bigcup_{q \in S} \delta_A(q, \alpha)$ (נשים לב כי S מצב של B, והוא גם תת קבוצה של Q_A).

הערה 1.22 מה זה אומר שL, M נתונות? אחת הדרכים הפורמליות היא ע"י אוטומט. לכן, אין בעיה להכריע שמילה בשפה, במידה ושפה נתונה ע"י אוטומט.

הערה 1.23 "זה קצת מבלבל" שאוסף המצבים של B הוא אוסף של קבוצות של מצבים מ־A. אפשר גם להגדיר את קבוצת המצבים של B בצורה הבאה:

$$\{q_S | S \subseteq Q_A\}$$

אבל אנחנו "צריכים להתרגל לדברים מבלבלים".

כדי להוכיח ש־$L(B) = L(A)$ נראה תחילה את למה העזר הבאה:

למה 1.24 לכל $S \subseteq Q_A$ ולכל $w \in \Sigma^*$, מתקיים[17]:

$$\delta_B^*(S,w) = \delta_A^*(S,w)$$

לפני הוכחת למת העזר, נראה כי למת העזר $\Leftarrow L(A) = L(B)$:

תהי $w \in \Sigma*$. $w \in L(A) \Leftrightarrow \delta_A^*(Q_0,w) \cap F_A \neq \phi \Leftrightarrow$ מלמת העזר $\delta_B^*(Q_0,w) = \delta_A^*(Q_0,w) \Leftrightarrow \delta_B^*\left(\overbrace{Q_0}^{=q_o}, w\right) \in F_B \Leftrightarrow w \in L(B)$.

נותר אם כך רק להוכיח את למת העזר: **הוכחה:** נוכיח באינדוקציה על $|w|$:

בסיס: אם $|w| = 0$:

$$\delta_B^*(S,\varepsilon) = S = \delta_A^*(S,\varepsilon)$$

נניח נכונות עבור מילים באורך n, ונוכיח עבור $|w| = n+1$: נרשום $w = u\alpha$, $|u| = n$. אזי:

$$\delta_B^*(S,u\alpha) = \delta_B(\delta_B^*(S,u),\alpha) \overset{ind.}{=} \delta_B(\delta_A^*(S,u),\alpha) \overset{def}{=} \delta_A^*(\delta_A^*(S,u),\alpha)$$
$$= \bigcup_{q \in \delta_A^*(S,u)} \delta_A(q,\alpha) = \delta_A^*(S,u\alpha)$$

■

■

מסקנה 1.25 REG סגור לפעולת השרשור, שכן הראינו שהשרשור של שתי שפות ב־REG ניתן לזיהוי ע"י אוטומט לא דטרמיניסטי, ומהמשפט מעלה אפשר לזהות את האוטומט הזה עם אוטומט דטרמיניסטי, כנדרש.

1.2 שקילות Myhill-Nerode

הגדרה 1.26 תהי L שפה מעל Σ, ויהיו $u, w \in \Sigma^*$.
נאמר ש־u, v שקולות ביחס ל־L, ונסמן $u \sim_L v$, אמ"מ לכל $w \in \Sigma^*$, $(uw \in L) \leftrightarrow (vw \in L)$.

נראה דוגמא:
$\Sigma = \{a\},\ L = \{a^n | n = 1\,(mod 5)\}$. נשאל, האם הן שקולות ב־$L$? $u = a^7,\ v = a^{22}$.
תחילה, נשים לב כי שתי המילים הללו אינם ב־L. אבל, ההגדרה כלל לא מתייחסת לזה.
$a^7 w \in L \Leftrightarrow |w| = 4\,(mod 5) \Leftrightarrow a^{22} w \in L$. על כן, u, v שקולות.

יחס זה הוא אכן יחס שקילות, אם כי לא נוכיח זאת (רפלקסיבי, סימטרי וטרנזיטיבי).
את מחלקת השקילות של מילה w ביחס לשפה L נסמן ב־$[w]_L$.

טענה 1.27 אם $u \sim_L v$ ו־$w \in \Sigma^*$, אזי $uw \sim_L vw$.

הטענה נכונה, כי $uwz \in L \Leftrightarrow vwz \in L$ לכל $z \in \Sigma^*$.
הכיוון השני[18]לא נכון - למשל, תהי L מעל $\Sigma = \{a,b\}$ אוסף המילים שבהן כל האותיות חוץ מהראשונה הן b, יהיו $u = ba, v = ab$. הן אינן שקולות, שכן אם נוסיף להן את המילה הריקה, $v \in L$ אבל $u \notin L$.
אם נביט ב־$w = aaa$, $uw \sim_L vw$, שכן לא משנה מהי המילה שנוסיף z, המילים שיתקבלו לא יהיו בשפה.

כתרגיל (נוכיח כמשפט בפעם הבאה) - הראה כי אם L שפה רגולרית, אזי מספר מחלקות השקילות הוא סופי. גם הכיוון ההפוך של טענה זו הינה נכון. זה נותן לנו תנאי הכרחי ומספיק להיות שפה רגולרית.

מכיוון וזהו יחס שקילות, נוכל להביט במחלקת השקילות המתאימה עבור $v \in \Sigma^*$:

$$[v]_L = \{u \in \Sigma^* | u \sim_L v\}$$

מתקיים: אם $u \sim_L v$, אזי:

$$[u]_L = [v]_L$$

ישירות מההגדרה של מחלקת שקילות.

דוגמאות

1. תהא L מעל $\Sigma = \{a, b\}$ שפת המילים w כך ש:

$$\#_a(w) = 1 \,(mod 3)$$

טענה 1.28

$$[bb]_L = \{w \in \Sigma^* | \#_a = 0 \,(mod 3)\}$$

הוכחה: תהי $u \in \Sigma^*$ כך ש:$\#_a(u) = 0 \,(mod 3)$. נראה הכלה דו כיוונית:

לכל מילה $w \in \Sigma^*$, אם $bbw \in L$, אזי $\#_a(w) = 1 \,(mod 3)$ (שכן ב־bb אין כלל a־אים). לכן $\#_a(uw) = 1 \,(mod 3)$, ולכן $uw \in L$.

מתקיים גם שאם $uw \in L$, אזי $bbw \in L$, בדיוק באותה הצורה. לכן:

$$\{u \in \Sigma^* | \#_a 0 \,(mod 3)\} \subseteq [bb]_L$$

כמו כן, אם $u \in [bb]_L$ ($u \sim_L bb$), אז בהכרח $\#_a(u) = 0 \,(mod 3)$, כי אחרת, מתקיים אחד משני הדברים הבאים:

- $\#_a(u) = 1 \,(mod 3)$: אזי, $u\varepsilon \in L$ אבל $bb\varepsilon \notin L$, בסתירה.
- $\#_a(u) = 2 \,(mod 3)$: אזי, $ua \notin L$ אבל $bba \in L$, בסתירה.

לכן בסה"כ מהכלה דו כיוונית, קיבלנו:

$$[bb]_L = \{w \in \Sigma^* | \#_a = 0 \,(mod 3)\}$$

כפי שרצינו להוכיח. ■

קל לראות כי בשפה זו ישנן בדיוק 3 מחלקות שקילות - המילים בהן מספר ה־a מתחלק בשלוש ללא שארית, עם שארית 1 ושארית 2.

2. תהי L מעל $\Sigma = \{a, b\}$ שפת המילים u שבהן $\#_a(u) > \#_b(u)$.

למשל, $aab \in L$, אבל $bba \notin L$.

מהן מחלקות השקילות ביחס $\sim_L$?

עבור שתי מילים $u, v \in \Sigma^*$, $u \sim_L v$ אמ"מ $\#_a(u) - \#_b(u) = \#_a(v) - \#_b(v)$.

דוגמא כללית:

יהא A DFA, $A = (Q, \Sigma, \delta, q_0, F)$. נגדיר לכל $q \in Q$:

$$L_q := \{u \in \Sigma^* | \delta^*(q_0, u) = q\}$$

[17] אין כאן בעיה, שכן מצד שמאל יש קבוצה יחידה של מצבים ב־A, ובצד ימין של קבוצה של מצבים ב־A.
[18] במקום "אזי" נשים "אמ"מ".

טענה 1.29 לכל $q \in Q$, אם $u, v \in L_q$, אזי $u \sim_{L(A)} v$.

הוכחה: תהא $w \in \Sigma^*$. אזי $uw \in L(A) \Leftrightarrow \delta^*(q_0, uw) \in F = \delta^*\left(\overbrace{\delta^*(q_0,u)}^{q}, w\right) = \delta^*(\delta^*(q_0,v),w) = \delta^*(q_0,vw) \Leftrightarrow vw \in L(A)$.

■

נשים לב כי הטענה <u>אינה</u> אומרת שכל מצב מגדיר מחלקת שקילות - יכול להיות שמחלקת שקילות מתקבלת מיותר ממצב אחד.
למשל, אם נביט באוטומט DFA בו יש כמה מצבים וכולם מקבלים, אזי כולם שקולים זה לזה.
עוד דוגמא: אוטומט בעל 6 מצבים מחוברים במעגל, ועוברים מאחד לשני כאשר נכנסת האות a, כאשר q_1, q_4 מקבלים. אזי:

$$L_{q_2} = \{a^n | n = 2\,(mod 6)\}\,,\ L_{q_5} = \{a^n | n = 5\,(mod 6)\}$$

במקרה הזה נקבל כי המילים בשפות הללו שקולות זו לזו ביחס לשפה של האוטומט $L(A)$.

מסקנה 1.30 אם L שפה רגולרית, אזי מספר מחלקות השקילות לפי $\sim_L$ הוא סופי. יתר על כן, אם A הוא DFA המקיים $L(A) = L$, אזי מספר מחלקות השקילות חסום ע"י מספר המצבים ב-A.

מסקנה זו בעצם נותנת לנו מבחן בנוגע להיות שפה רגולרית - ראינו בשפה בדוגמא 2 כי מספר מחלקות השקילות הוא אינסופי, ולכן שפה זו לא יכולה להיות רגולרית.
בשיעור הקודם ראינו כי אם A הוא DFA, ו-$\delta^*(q_0,w) = \delta^*(q_0,w')$, אז $w \sim_{L(A)} w'$, ולכן אם A אוטומט בעל t מצבים, אז ב-$L = L(A)$ יש לכל היותר t מחלקות שקילות $M.N.$.
היום נראה את המשפט ההפוך:

משפט 1.31 אם L מעל Σ היא בעלת $t < \infty$ מחלקות שקילות עבור $\sim_L$, אזי קיים אוטומט A DFA בעל t מצבים כך ש: $L(A) = L$.

מסקנה 1.32 $L \in REG$ אמ"מ היחס $\sim_L$ מגדיר מספר סופי של מחלקות שקילות.
יתרה מזו, משפט זה נותן מספר מינימלי t של מצבים באוטומט שמזהה את L, שכן מהמשפט הקודם, יש t מילים שכשהאוטומט ירוץ עליהן נגיע למצבים שונים זה מזה, ולכן לא ניתן לבנות אוטומט שיזהה את השפה שיהיה בעל מספר מצבים קטן מ-t.

נוכיח את המשפט: **הוכחה:** נגדיר אוטומט A DFA באופן הבא: $A = (Q, \Sigma, \delta, q_0, F)$ כאשר:

- Σ הוא הא"ב שמעליו L מוגדרת.
- Q - ב-Q יהיה מצב q_M לכל מחלקת שקילות M ביחס ל-$\sim_L$.

נרצה להגדיר את δ, q_0, F כך שיתקיים:

$$\delta^*(q_0, w) = q_{[w]_L}$$

- נגדיר את $q_0 = q_{[\varepsilon]_L}$.
- F - נשים לב שלכל מחלקת שקילות M, או שכל המילים בה שייכות ל-L, או שכל המילים ב-M אינן ב-L.

$$F = \{q_M \in Q | \text{all the words in M are in L}\}$$

- δ: תהא M מחלקת שקילות ותהא $\alpha \in \Sigma$. תהא $w \in M$, ונגדיר:

$$\delta(q_M, \alpha) = q_{[w\alpha]_L},\ q_M = q_{[w]_L}$$

יש לבדוק שהרשום מעלה מוגדר היטב, שכן ההגדרה תלויה בנציג w שבחרנו - ו-δ אכן מוגדרת היטב, שכן אם $w' \in M$ מילה אחרת, אזי $w \sim_L w'$, ולכן $w\alpha \sim_L w'\alpha$ (אף הוכחנו זאת), כלומר $[w'\alpha]_L = [w\alpha]_L$, ועל כן:

$$q_{[w\alpha]_L} = q_{[w'\alpha]_L}$$

נותר להוכיח כי $L(A) = L$.

טענה 1.33 לכל $w \in \Sigma^*$,

$$\delta^*(q_0, w) = q_{[w]_L}$$

מסקנה 1.34 $w \in L(A) \Leftrightarrow \delta^*(q_0, w) \in F \overset{claim}{\Leftrightarrow} q_{[w]_L} \in F \Leftrightarrow$ כל המילים ב־$[w]_L$ שייכות ל־$L \Leftrightarrow w \in L$, ולכן $L(A) = L$.

הוכחת טענת העזר נותרת לקורא השקדן - באינדוקציה על אורך המילה w. ■

המצב שהגענו אליו הוא די אוטופי - הגדרנו מודל חישובי - אוטומט, ואז הגדרנו מודל שקל יותר לעבוד איתו - DFA (לפחות עם חלק מהדברים), ומודל יותר חזק NFA, הראינו שכל דבר שאפשר לעשות עם NFA אפשר לעשות גם עם DFA, ולסיום הגדרנו יחס שקילות שנותן לנו זיהוי מלא האם שפה היא רגולרית או לאו. כמו כן, מצאנו בדיוק את מספר המצבים המינימלי ל־DFA של השפה. בתרגול אף ראינו אלגוריתם למציאת האוטומט המינימלי ומחלקות השקילות.
אנחנו נראה בהמשך שבמודלים מורכבים יותר, אפילו קצת יותר מורכבים מ־DFA, אין קריטריון מפורש כפי שמצאנו כאן.
המשפט אמ"מ נקרא משפט מייהיל נרוד, והם הגדירו את יחס השקילות על מנת להוכיח את המשפט.

1.3 ביטויים רגולריים

עד כה הגדרנו שפה ע"י האוטומט שלה. עם זאת, דרך הייצוג הזו לא נתנה לנו דרך לבנות מילים בשפה נתונה.
באופן כללי, ישנן שתי דרכים לייצוג שפה:

- מודל חישובי - אוטומט שמזהה את השפה
- דקדוק - בדר"כ נותן לנו דרך לבנות מילים בשפה.

דרך דקדוקית שנכיר היא ביטויים רגולריים.

הגדרה 1.35 ביטוי רגולרי מעל א"ב Σ יוגדר בצורה הבאה:

1. הביטוי $r = "\phi"$ מתאר את השפה $L_r = L_{"\phi"} = \phi$
2. הביטוי $r = "\varepsilon"$ מתאר את השפה $L_r = L_{"\varepsilon"} = \{\varepsilon\}$.
3. עבור $\alpha \in \Sigma$, הביטוי $r = "\alpha"$ מתאר את השפה $L_{"\alpha"} = \{\alpha\}$.
4. עבור r_1, r_2 ביטויים רגולריים מעל Σ:
   (א) הביטוי $r = (r_1)(r_2)$ מתאר את השפה $L_r = L_{r_1} \circ L_{r_2}$.
   (ב) הביטוי $r = (r_1) \cup (r_2)$ מתאר את השפה $L_r = L_{r_1} \cup L_{r_2}$.
   (ג) הביטוי $r = (r_1)^*$ מתאים ל־$L_r = (L_{r_1})^*$[19].
5. הביטוי $r = (r_1)^+$ מתאר את $L_r = L_{r_1} \circ (L_{r_1})^*$.

[19] נזכיר כי:

$$L^* = \{w | \exists k \in \mathbb{N} \text{ and } x_1, ..., x_k, \ \forall 1 \leq i \leq k, x_i \in L, \text{ and } w = x_1, ..., x_k\}$$

דוגמאות יהי $\Sigma = \{a, b\}$.

- $r = (a) \cup (\varepsilon)$ מגדירה את השפה שמכילה את a, ε.
- $(b)^+$ מגדירה את השפה של המילים שהן רצף של האות b.
- עבור $r = (a) \cup (\varepsilon) \cdot (b)^+$, $L_r = \{w | \#_b(w) \geq 1, \text{ and all letters are b other than the first letter } \}$. בדר"כ נכתוב זאת בצורה הבאה:

$$(a \cup \varepsilon) b^+$$

נשאלה השאלה, מדוע אין סימן $\wedge$ בדקדוק שלנו, כאשר עבור $r = (r_1) \wedge (r_2)$, $L_r = L_{r_1} \wedge L_{r_2}$?
כדי ליצור מילה כזו, היינו צריכים למצוא מהי השפה שמתארת r_1, השפה שמתארת את r_2, ואז להבין איך נראה ביטוי בשפה הנ"ל, וזוהי לא בנייה קונסטרוקטיבית כמו בהגדרה המקורית שלנו. מתוך ההגדרה המקורית אנו כבר יודעים איך לבנות ישירות מילה בשפה מבלי להסתבך.

נרצה להראות כי הכלי שהגדרנו חזק, דהיינו, אפשר לתאר בעזרתו הרבה שפות.
ראינו בתרגול: לכל ביטוי רגולרי r מעל Σ מתקיים $L_r \in REG$. נרצה להראות את הכיוון השני:

משפט 1.36 תהי L שפה מעל Σ רגולרית. אזי יש ביטוי רגולרי r כך ש־$L_r = L$.

הוכחה: נגדיר "אוטומט ביטויים רגולריים"[20] באופן הבא: אוטומט כזה A הוא חמישייה $A = (Q, \Sigma, \delta, q_0, F)$, כאשר Q, Σ, q_0, F כמו באוטומט רגיל, ו־δ היא אוסף של שלשות (q_1, r, q_2) כאשר $q_1, q_2 \in Q$, ו־r הוא ביטוי רגולרי מעל Σ.

שתי דוגמאות:

- נביט ב: $r = (a \vee \varepsilon) \cdot b^+$. אוטומט ביטויים רגולריים שמתאים לו יהיה:
ציור
כאשר המצבים הם q_0, q_1, q_1 מצב מקבל, ומעל המעבר ביניהם מופיע $(a \cup \varepsilon) \cdot b^+$, כלומר אפשר לעבור בין המצב הראשון לשני אם המילה היא מהצורה הזו. במקרה הזה, $\delta = \{(q_0, (a \cup \varepsilon)) b^+, q_1\}$.
- אפשר לצייר אוטומט גם בצורה שונה ־ המצבים כמו בקודם, אולם מ־q_0 יוצאים שני חצים ל־q_1 ־ מעל הראשון מופיע הביטוי $a \cdot b^+$, ומעל השני b^+.
כל אוטומט כזה מקבל בדיוק את השפה שמתאימה לביטוי.

הערה 1.37 נרצה למצוא דרך לצאת מאוטומט רגיל, בעזרת סדרה של אוטומטים ביטויים רגולריים להקטין את המצבים, עד שנגיע לאוטומט בעל שני מצבים ־ כאשר מעל החץ למצב המקבל יופיע הביטוי הרגולרי המתאים לשפה.

נגדיר ריצה חוקית על מילה $w \in \Sigma^*$ להיות סדרה:

$$q_0 \overset{u_1}{\underset{r_1}{\rightsquigarrow}} q_1 \overset{u_2}{\underset{r_2}{\rightsquigarrow}} \ldots \overset{u_k}{\underset{r_k}{\rightsquigarrow}} q_k$$

כאשר המצב הראשון ־ q_0, ובכל מעבר מהצורה $q_i \overset{u_{i+1}}{\underset{r_{i+1}}{\rightsquigarrow}} q_{i+1}$ מתקיים:

- $u_{i+1} \in L_{i+1}$.
- $(q_i, r_{i+1}, q_{i+1}) \in \delta$.
- $u_1 \cdot u_2 \cdot \ldots \cdot u_k = w$.

הריצה היא מקבלת אם q_k הוא מצב מקבל.

[20] הייבריד של אוטומט וביטויים רגולריים, שכן לא נוכל לעבור מיד מהאוטומט של השפה לביטוי רגולרי.

נביא דוגמא, ונקווה שנדע להכליל אותה להוכחה פורמלית:
ציור
תחילה, נמצא קשתות מקבילות - מתחילות באותה צומת ומגיעות לאותה הצומת. נרצה להחליפן בקשת שמתארת את איחוד הביטויים - נחליף את שתי הלולאות בציור הראשוני בלולאה עם הביטוי $(a \cup b)$.
פעולה נוספת - להיפטר מצומת. קל להפטר מצמתים שאין עליהן לולאות. נביט על קשתות שנכנסות לצומת ויוצאות מהצומת. נמחק הצומת (מצב) ונכתוב קשתות נוספות המתארות את המעבר - בציור מחקנו צומת והוספנו שני מעברים מהמצב הראשוני - מעבר שמעליו מופיע ab, ומעבר שני עם הביטוי aa.
נמשיך לבטל צמתים. נביט בצומת שאינו מקבל - נביט בכל הדרכים להכנס אליו ע"י קשת ולצאת מהן ע"י קשת. אם היתה קשת שיוצאת מהצומת ע"י הביטוי r, היינו מוסיפים קשתות מקוצרות עם השרשורים. יש בצומת שאנו מביטים בה לולאה, לכן נחליף ב-$aa\left(a \cup b\right)^* r$.
אבל לצומת הזו אין קשת שיוצאת ממנה, ולכן אין בעיה פשוט למחוק אותו - שכן אם מילה מגיעה לצומת הזו, היא לא מקבלת.
נשאר עם צומת כניסה עם קשתות שיוצאות ממנה לשני מצבים מקבלים בעלי לולאות. הלולאות מהוות בעיה - כיצד נטפל בה? נמחק את הלולאה, ובכל קשת שנכנסת למצב המקבל נוסיף את הביטוי שהיה בלולאה עם כוכבית עליו - במקרה שלנו $ab\left(b\right)^*$.
נשארנו עם מצב התחלתי ואוסף קשתות שמחברת אותו למצבים מקבלים. אפשר לחבר את שתי הקשתות לאותו מצב מקבל ולמחוק את השני. לכן נותרנו עם אוסף קשתות מקבילות. עם מצב זה אנו יודעים להתמודד - נחליף את אוסף הקשתות בקשת אחת עם "או" בין הביטויים שעל הקשתות, ונסיים.

■

2 מכונת טיורינג

עד כה, הצלחנו להבין מיהי שפה רגולרית, הכרנו DFA, הוספנו חוסר דטרמיניזם, ראינו וריאציות שונות שתמיד יצאו שקולות להגדרה של שפה רגולרית, הכרנו דקדוק של שפה...
נרצה למצוא מודל חישובי שכן יוכל לזהות שפות אחרות, כמו למשל a^nb^n שאינה רגולרית, אך מאוד פשוטה. הגדרה פורמלית של שפה זו:

$$L = \{a^nb^n | n \in \mathbb{N}\}$$

מה DFA עשה?
קיבל מילה - רצף של אותיות, ולפי המצב הנוכחי והאות הבאה החליט לאיזה מצב לעבור.
שאלנו, אילו חיזוקים אפשר להוסיף למודל הזה? ראינו כבר הוספת אי דטרמיניזם, אבל זה לא שינה את משפחת השפות שזיהינו:

1. דו־צידיות: כלומר, אפשרות לקרוא לא רק משמאל לימין אלא גם מימין לשמאל (כלומר אחורה). במקרה שכזה יהיה אפשר לבסס את בחירת המצב הבא לפי אותיות קודמות (פשוט ללכת אחורה ולבדוק מה היה). לכאורה, מצב כזה יכולנו לפתור עם אוטומט רגיל ע"י "זיכרון" של האות הרלוונטית. אזי, לא ברור שתוספת זו מוסיפה כוח למודל (בחלק מהתרגולים הראינו שזה לא מוסיף).
כדי לעשות את זה, פשוט על כל קשת נוסיף כיוון.

2. זיכרון:

(א) ע"י הוספת מחסנית - על הקשתות נכתוב (מלבד את האות שמובילה למצב הבא) גם אות נוספת, וכן H בשביל $push$ ו־p בשביל pop. אם כתוב אות ואז H, נעבור למצב הבא, ולמחסנית נדחוף את האות שהיתה כתובה. לא נגדיר את המודל הזה עד הסוף, כאן נחליט שבחלק מהמצבים מותר לנו לא לעשות $push$ ולא לעשות pop. אם כתוב pop, נעשה pop, ונוכל להמשיך אך ורק אם האות שיצאה היא אכן האות שכתבנו על הקשת. אוטומט שכזה נקרא "אוטומט מחסנית", והוא חשוב "לכל מני דברים". נוסיף סולמית למחסנית כדי שנוכל לזהות כאשר המחסנית מתרוקנת.
בעזרת אוטומט זה אפשר לזהות את השפה a^nb^n: נעשה $push$ ל־a עד שנגיע ל־b, ובכל פעם שנגיע ל־b נעשה pop. אם נראה a אחרי b האוטומט לא יקבל. כדי לוודא שמספר ה־a שווה למספר ה־b נעשה pop עד סוף המחסנית, אם יש יותר מידי b לא נוכל לעשות pop שנצטרך, ואז נדע לא לקבל את המילה, אם יש פחות מידי b לא נרוקן את המחסנית - נבדוק בסוף הריצה אם pop נוסף נותן סולמית, אם יש יותר מידי b יעשה pop ל־a.
נשאלת השאלה, האם אפשר לזהות בעזרת אוטומט מחסנית את השפה $a^nb^nc^n$? לא בעזרת אוטומט מחסנית (שהיא לא ממש מסובכת), אלא בעזרת אוטומט מחסניות - אוטומט עם 2 מחסניות (יותר על סוג אוטומט זה בהמשך). אזי, אוטומט מחסנית חזק מהאוטומט הרגיל שלנו, אבל לא בהרבה. נצטרך להוסיף לו משהו נוסף:

i. אוטומט מחסנית + אי־דטרמיניזם - מכל מצב ניתן להוסיף כמה קשתות, אפילו אם יש עליהן את אותה האות. מסתבר שאוטומט זה חזק יותר מאוטומט מחסנית בלבד. בעזרת אוטומט זה אפשר לזהות משפחת שפות בשם CFL. עם זאת, השפה $a^nb^nc^n \notin CFL$. מצד שני, כל שפות התכנות הן ב־CFL, לכן המשפחה הזו חשובה - אי אפשר להבין קומפילציה מבלי להבין משפחה זו, ויש לאוטומט זה כח מסויים (בדר"כ מדברים על משפחה זו ביתר אריכות, אולם השנה בשל השביתה נאלץ לוותר).

(ב) סרט דו צידי R/W: לתת לאוטומט לרשום מחדש על האותיות של המילה עצמה. אפשר לתת לו גם לרשום על אותיות אפילו הן לא של המילה עצמה - כלומר לפני, אחרי, וכו'. אם אנחנו לא יכולים לחזור אחורה עם זאת אנחנו למעשה לא יכולים לקרוא, ואז לא שיפרנו כלום.

הגדרה 2.1 (לא פורמלית) מכונת טיורינג היא אוטומט שמוסיפים לו דו־צידיות וסרט דו צדדי R/W. אז אפשר כאמור לכתוב וגם לקרוא את מה שכתבנו.

נחליט שבמכונת טיורינג גם מותר לנו להשאר במקום. על הצלעות נכתוב את האות, L/R, ואות שנכתוב - אם לא נרצה לכתוב על, פשוט נכתוב את אותה האות שנקראה. כמו כן נחליט שבסוף הקלט יש לנו תו בעזרתו נוכל לזהות שהקריאה הסתיימה.
נשאלת השאלה, אילו שפות ניתן לזהות בעזרת מכונת טיורינג?

- REG - נקבע מעברים רק שמאלה, וכשנסיים לקרוא את המילה, בהתאם למצב שהגענו אליו נקבע אם לקבל או לדחות את המילה.
- a^nb^n - נקרא a, נמחק אותה, נלך קדימה עד שנמצא b, אם מצאנו אחד נמחוק אותו, ונמשיך בתהליך זה, וכך נוכל לזהות אם יש יותר b מ־a או להפך.

- למעשה מכונת טיורינג יכולה לזהות כל שפה (שאינה משתמשת באנרגיה אינסופית?!) - היא יכולה לעשות כל דבר שמחשב יכול לעשות.

בשלב הבא נגדיר את המודל באופן פורמלי, ונדבר על עוד דברים שהמודל הזה יכול לעשות, ונשתכנע שהוא אכן יכול לעשות את כל הדברים האלו. 18/4/2012

נגדיר את מכונת טיורינג בצורה פורמלית:

הגדרה 2.2 מכונת טיורינג היא שביעיה $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, כאשר:

- Q - קבוצת מצבים סופית, $q_0, q_{acc}, q_{rej} \in Q$, $q_{acc} \neq q_{rej}$.
- Σ - א"ב הקלט, סופי, ולא מכיל את הסימן _.
- Γ - א"ב העבודה (הסרט), $\Sigma \subseteq \Gamma$, $\_ \in \Gamma \backslash \Sigma$.
- δ - פונקציית המעברים:

$$\delta : (Q \backslash \{q_{acc}, q_{rej}\}) \times \Gamma \to Q \times \Gamma \times \{R, L\}$$

פונקציית המעברים מתוארת באמצעות טבלה.

הגדרה 2.3 קונפיגורציה של מכונת טיורינג $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ היא רצף $x\,(q, \alpha)\,y$ כאשר:

- $q \in Q$ - המצב הנוכחי של האוטומט.
- $\alpha \in \Gamma$.
- $y \in \Gamma^*$ מחרוזת שהאות האחרונה בה (אם קיימת, כלומר אפשר ש-y היא מחרוזת ריקה) איננה _.
- $x \in \Gamma^*$ מחרוזת שהאות הראשונה בה (אם קיימת) אינה _.

הגדרה 2.4 בהנתן קונפיגורציה $x\,(q, \alpha)\,y$, הקונפיגורציה הבאה מחושבת כדלהלן:

1. אם $q \in \{q_{rej}, q_{acc}\}$, אז זו קונפיגורציה סופית, ואין לה קונפיגורציה עוקבת.
2. נכתוב $\delta\,(q, \alpha) = (q', \beta, d)$ ($d \in \{R, L\}$).

(א) אם $d = R$, הקונפיגורציה הבאה היא $x\beta\,(q', \gamma)\,z$, כאשר אם y אינה ריקה, נכתוב $y = y_1 z$, ונגדיר $\gamma = y_1$. אם $y = \varepsilon$, נגדיר $\gamma = \_$, $z = \varepsilon$.

(ב) אם $d = L$, הקונפיגורציה הבאה היא $x\,(q', \gamma)\,\beta z$ - "נפעל באופן דומה רק הפוך".

נביט בדוגמא הבאה:

δ	a	b	_
q_0	$q_0, \#, R$	$q_0, \#, R$	$q_0, \#, R$

כל קונפיגורציה שנגיע אליה לא תהיה סופית - תמיד נוכל להמשיך הלאה!

כמו כן, היא לא חוזרת על אף קונפיגורציה פעמיים, כי לאחר זמן t הראש נמצא אחרי t #-אים, כלומר לאחר זמנים שונים נקבל מספר שונה, על כן הקונגפוגורציות אינן זהות.

הגדרה 2.5 ריצה חלקית של M היא סדרת קונפיגורציות סופית $C_1, C_2, ..., C_k$ שבה כל קונפיגורציה היא עוקבת של קודמתה.

הגדרה 2.6 נגיד שריצה חלקית היא ריצה מלאה אם בקונפיגורציה האחרונה C_k, q הוא q_{rej} או q_{acc}. אם אין ריצה מלאה אשר הקונפיגורציה הראשונה בה היא C_1, נגיד ש-M אינה מסיימת את ריצתה כאשר הקונפיגורציה הראשונה שלה היא C_1.

הגדרה 2.7 ריצה חלקית של M על קלט x היא ריצה חלקית שבה הקונפיגורציה הראשונית היא $\varepsilon\,(q_0, x_1)\,x_2 \cdot ... \cdot x_{|x|}$ (אם $x = \varepsilon$, אז $\varepsilon\,(q_0, \_)\,\varepsilon$).

אתנחתא:
נדבר מעט על הציונים של הבחנים. "הם סבירים, אפילו קצת יותר מידי גבוהים לטעמי", יש הבדלים לפי הגרסאות של הבחנים. הצוות מודע לזה. כרגע לא יעשה שום שינוי לציון. אם יהיה צורך בסוף, אז נראה מה יעשה.

הגדרה 2.8 נגיד ש־M מקבלת מילה $x \in \Sigma^*$ אם יש לה ריצה מלאה על x, וריצה זו מסתיימת בקונפיגורציה שבה מצב האוטומט הוא q_{acc}.
M דוחה $x \in \Sigma^*$ אם יש לה ריצה מלאה על x, וריצה זו מסתיימת בקונפיגורציה שבה מצב האוטומט הוא q_{rej}.

הערה 2.9 מההגדרה מעלה אנו מקבלים כי יש מילים שאינן מתקבלות ואינן נדחות. זה מעלה בפנינו בעיה ־ איך נסמן את השפה של M? או, מה נעשה עם המילים האלו ש"נופלות באמצע"?
אחד הדברים שנוכיח בשיעורים הקרובים הוא שאם מכונה לא עוצרת בריצה על מילה מסויימת בשלב כלשהוא, אנחנו לא יודעים אם היא תעצור בכלל (יכול להיות שהיא תעצור בשלב הבא, עוד שניים... או שלא תעצור בכלל).

הגדרה 2.10 השפה $L \subseteq \Sigma^*$ המזוהה ע"י המכונה M היא אוסף המילים x שאותן M מקבלת. מסמנים אותה $L(M)$.

הגדרה 2.11 נגיד ש־M מכריעה את $L(M)$ אם לכל קלט $x \in \Sigma^*$, M מקבלת או דוחה את x (כלומר, אם לכל קלט $x \in \Sigma^*$ יש ריצה מלאה).

הערה 2.12 נשים לב כי מן ההגדרה, M יכולה להכריע רק את השפה שהיא מקבלת ־ דהיינו $L(M)$.

כשדנו באוטומטים, שאלנו, איזו משפחה מזוהה ע"י האוטומטים? כאן יש לנו שתי שפות רלוונטיות:

הגדרה 2.13 R היא משפחת השפות הניתנות להכרעה ע"י מכונת טיורינג.

הגדרה 2.14 RE היא משפחת השפות הניתנות לזיהוי ע"י מכונת טיורינג.

R=Recursive, RE=Recursively Enumerable, מדוע? נראה את זה בהמשך.

הערה 2.15 מההגדרות ברור כי $R \subseteq RE$.

שאלה מעניינת: $RE \overset{?}{\subseteq} R$.
בשיעורים הקרובים נראה כי אין הכלה בכיוון השני, דהיינו קיימות שפות ב־RE שאינן ב־R.
שאלה נוספות: הם קיימות שפות L, $L \notin RE$?
תשובה: כן! עוצמת קבוצת מכונות הטיורינג היא בת מניה. מדוע? כל מכונת טיורינג אפשר לתאר בעזרת מחרוזת סופית של הא"ב העברי (פלוס אולי הספרות העשרוניות). כמה מחרוזות כאלו יש? מספר בן מניה. לכן עוצמת קבוצת מכונות הטיורינג היא בת מניה.
מצד שני, בשביל לתאר שפות יש צורך במספר אינסופי של ביטים, לכן מספר השפות אינו בן מניה ־ בגודל $\aleph$.
זהו הנושא בו נעסוק בשיעורים הקרובים.
בתור התחלה, נרצה לדבר על השפות המעניינות ב־R:

- $a^n b^n \in R$, $a^n b^n c^n \in R$.
-

$$\left\{x\#y\#z | x,y,z \in \{0,1\}^*,\ x+y=z\right\}$$

מההגדרה של x, y, z אפשר לחשוב עליהם כמספרים בינאריים. האם השפה הזו ב־R?
כן! נזכור שיחסית קל להוסיף 1 למספר בינארי: למשל, כדי להוסיף 1 ל־x, נקרא אותו עד #, ואז נתחיל לחזור שמאלה ־ כל פעם שרואים 1 מחליפים אותו ב־0, עד שרואים 0 ומחליפים אותו ב־1. אם לא רואים 0 מחליפים את ה־_ ב־1.
להוריד 1 מ־y זה גם אפשרי ־ בצורה דומה והפוכה ־ מחפשים את ה־1 הכי ימני ב־y, הופכים אותו ל־0, והופכים את השאר... רק צריך לוודא ש־y אינו אפס.
כך אפשר לחבר את x ו־y: לחבר ל־x אחד עד ש־$y = 0$. כלומר, אפשר לחבר במכונת טיורינג.
אפשר גם לבצע כפל ־ לכתוב עותק (או כמה עותקים) של x משמאל ל־x, מוסיפים את x ל־x, ואז מורידים ל־y אחד, מעתיקים את האיקס למקום המקורי, ושוב נחבר. אז נוריד אחד מ־y, נשלים את x למה שהיה, נחבר שוב, וחוזר חלילה.
העלאה בחזקה גם כן אפשר לעשות.

- מה עם הבעיה הבאה:
נתונה מערכת משוואות לינאריות, והשמה למשתנים במערכת. צריך לקבל את המערכת יחד עם ההשמה, אמ"מ ההשמה מהווה פתרון למערכת המשוואות. האם ניתן לזהות את השפה הזו? נענה על השאלה הזו בפעם הבאה.

- ומה עם בעיית $3-SAT$?

23/4/2012

אפשר לדבר גם על מכונת טיורינג עם שני סרטים ושני ראשים קוראים. אז פונקציית ה־δ מקבלת שתי אותיות של קלט - אחד עבור הקלט שעליו הראש הראשון, השני על השני. אפשר לדבר על מכונה על k סרטים. אפשר לדבר על מכונה עם שני סרטים אולם אחד מהם רק קריאה, השני הוא סרט העבודה עליו אפשר לקרוא ולכתוב.

יש הרבה מאוד ורסיות של מודלים של מכונת טיורינג. אחד מהם (נראה בתרגול) שיש סרט אחד אבל הוא דו כיווני (?). לכל מודל כזה אפשר למצוא את אוסף השפות הניתנות לזיהוי על ידו, ואוסף השפות המוכרעות על ידו.

הדבר היפה הוא, שהמודלים לא משנים את אוסף השפות המזוהות ולא את אוסף השפות המוכרעות!

זוהי תוצאה מפתיעה למשל, במעבר ממכונה בעלת שני סרטים למכונה בעלת סרט אחד - במקרה זה למשל אפשר לתכנן את המכונה החדשה כך ש"תחלק" את הסרט לשניים - למשל כל התאים האי־זוגיים יכילו את המידע של סרט אחד, והתאים הזוגיים יכילו את המידע של הסרט השני. נראה מסובך לראות מה עושים במקרה זה לגבי קריאת הנתונים (קריאת שתי אותיות לעומת קריאה של אחת), אבל ניתן לסמלץ גם את זה - נריץ על הסרט עד שנמצא את המיקום של הראש הראשון, ואז נריץ שוב עד שנמצא את המקיום של הראש השני, ואז ניתן לתכנן בהתאם.

כלומר, למרות שלמדנו רק הגדרה אחת, עם הגדרות אחרות היינו מקבלים את אותן המשפחות R, RE, ויתרה מזו - הסימלוץ גורר תשלום, אולם הוא פולינומיאלי (גם לגבי המקום וגם לגבי זמן הריצה). הרבה מהדוגמאות הללו נראה בתרגולים.

מכונות טיורינג גם יכולות לחשב פונקציות של הקלט:

1. מספר עוקב, מספר קודם.

2. חיבור, כפל, חזקות (גם ברציונליים - מכיוון ומכונת טיורינג היא גמישה מאוד, יש הרבה דרכים - למשל בעזרת חזקות של בינאריים למשל עם תו מפריד. רב הדרכים הסבירות להצגה "יעלו" בערך אותו הדבר - יהיה אפשר לבנות מכונת טורינג שמסמלצת פורמט אחד לשני).

בעיית השורש המשותף של פולינומים: קלט: משתנים $x_1, ..., x_n$, ומשוואות מהצורה:

$$x_1^2 + x_2 \cdot x_3^5 - 7 = 0$$

$$\vdots$$

שאלה: האם יש השמה $x_1, ..., x_n \in \mathbb{Z}$ כך שכל המשוואות מתקיימות?

השאלה הזו שקולה להגדרת שפת המשוואות שיש להן פתרון שלם - השאלה: האם השפה הזו ניתנת לזיהוי באמצעות מכונת טיורינג?

תחילה, האם זו בכלל שאלה שניתן לתת למכונת טיורינג כפלט? כדי שזה יהיה נכון, נצטרך דרך הצגה של משוואות מעל n משתנים. בה"כ מדובר במערכת מעל משתנים שלמים עם מקדמים שלמים.

זה אפשרי! תחילה יש להגיד למכונה כמה משתנים יש - אפשר לתת את המספר הזה כקלט.

כיצד נעביר את המשוואות? נגדיר סימן המפריד בין משוואות שונות.

כל משוואה היא אוסף של מונומים עם מקדמים. נותר להגיד איך מייצגים מונום - וכל אחד כזה מכיל מקדם, ומשתנים בחזקות שונות - לכן ניתן לייצג אותו כסדרה של $n+1$ של מספרים שלמים - הראשון הוא המקדם, n האחרים הם המקדמים של המשתנים.

על כן, זוהי דרך להעביר את מערכת המשוואות למכונת טיורינג.

עכשיו נשאל, האם זו שפה הניתנת לזיהוי ע"י מכונת טיורינג? כן!

הפתרון: לעבור על כל האפשרויות, אם נמצאה השמה המכונה תחזיר "כן", אחרת תחזיר לא.

כיצד עושים זאת? ישנן הרבה דרכים. לדוגמא, אפשר קודם כל לעבור על כל ההשמות בהן $x_1, .., x_n \in \{-1, 0, 1\}$, לאחר מכן נעלה מונה כלשהוא באחד ונעבור על כל ההשמות כך ש־$x_1, .., x_n \in \{-2, -1, 0, 1, 2\}$, וכן הלאה.

על כן, השפה הזו אכן ב־RE. נשאל, האם השפה הזו ב־R? שאלה זו נשארת כרגע בסימן שאלה.

השאלה הזו לא נוסחה כך, אבל בעצם נשאלה בשנת 1900 בגרסא אחרת ע"י הילברט - זוהי בעצם השאלה העשירית של הילברט: האם ניתן להכריע אם יש או אין פתרון למערכת משוואות פולינומיות?

שאלה זו ושאלות דומות הן הסיבה להמצאת מכונות טיורינג.

עוד שאלה אפילו עוד יותר חשובה שהטרידה את הילברט ורבים לפניו (ולא מעטים אחריו) שהיא אולי הבסיס של המתמטיקה - המספרים הממשיים הם קבוצה, אבל כל מספר ממשי בעצמו ניתן לייצג כקבוצה של רציונליים, כל רציונלי הוא בעצם זוג של שלמים, כל שלם אפשר לייצג כקבוצה של זוגות של טבעיים, וטבעיים גם ניתן לייצג ע"י קבוצות של קבוצות ריקות. כלומר, כולם בנויים על תורת הקבוצות בלבד. ישנן אוסף האקסיומות של המתמטיקה - $ZF(C)$. כל מה שניתן להוכיח מהאקסיומות הוא משפט, וכל מה שניתן להוכיח שאינו נכון מהאקסיומות הוא משפט שהשלילה שלו נכונה.

1. האם אוסף המשפטים הנובעים מ־$ZF(C)$ היא ב־RE? כלומר, האם אפשר להכניס את משפט למכונת טיורינג והיא תגיד אם הוא נכון.
2. האם אוסף המשפטים הללו הוא ב־R? כלומר, האם אפשר למצוא מכונה שתעצור גם אם המשפט לא נכון ותגיד שאינו נכון.
3. שאלה דומה אך שונה באופיה - האם לכל משפט p מתקיים $p \dashv ZF(C)$ או $\neg p \dashv ZF(C)$ (כלומר האם $ZF(C)$ שלמה?).

אבחנה: אם השאלה לשלוש ולאחד היא כן, אזי התשובה לשתיים היא כן - זאת כי אם המשפט אינו נכון, אזי המכונה יכולה למצוא אם המשפט לא נכון: אפשר להריץ שתי מכונות המוצאות הוכחות בו זמנית, לאחת מהן להזין את המשפט, לשניה את שלילת המשפט, ואז אחת מהן תסיים לפני השניה, ולפיה נדע מהי התשובה הנכונה.

התשובות לשאלות:

התשובה לשאלה העשירית של הילברט איננה ב־R (יכול להיות שלא נדע שהמערכת אינה פתירה). לגבי שלוש השאלות הנוספות שהצגנו:

1. כן - לפי המערכת של פייגה (עוד על כך בקורס "לוגיקה מתמטית") יש אלגוריתם לבדיקת נכונות של הוכחה. פשוט נרוץ על כל הוכחה אפשרית, וכך נדע עם המשפט ב־RE (ברגע שיש הוכחה לוקח זמן פולינומי לבדוק את נכונותה).
2. לא! בעצם לכל מערכת לוגית עשירה מספיק אי אפשר להכריע מה היא משפחת השפות המוכרעות על ידה.
3. לא! לכל מערכת אקסיומות עשירה מספיק (וקונסיסטנטית) יהיו טענות שלא נובעות ממנה, וגם שלילתן אינה נובעת ממנה.

מי שהוכיח לראשונה את התשובות לשאלות $2, 3$ הוא גדל.

טיורינג ניסח את השאלות הללו במובן של R, RE, ואז בעזרת מכונת טיורנג ענה עליהן. בשיעור הבא נוכיח על איזשהן בעיות שאינן ב־R.

2.1 מכונת טיורינג כקלט - קידוד של מכונת טיורינג

נחשוב על אלגוריתם כמכונת טיורינג - במשך הזמן נראה כי באמת כל אלגוריתם בסיסי שנחשוב עליו ניתן לממש כמכונת טיורינג.

אולם כעת, נראה כיצד ניתן לקודד מכונת טיורינג כדי שתוכל להתקבל כקלט ע"י מכונת טיורינג אחרת D. נסמן:

$$M = (Q, \Sigma, \Gamma, \delta, ...)$$

מהי הבעיה? הא"ב של המכונה M יכול להיות הרבה יותר גדול מזה של D, ואז מה נעשה כאשר הראש הקורא של M הוא על אות שאיננה ב־D?

התשובה עם זאת די פשוטה - מה שחשוב הוא למעשה מספר האותיות:

נחשוב על Σ ועל Γ כעל שני מספרים טבעיים - מהן האותיות עצמן זה משנה. נחליט שכאשר יש למשל 10 אותיות ב־Σ, הן $1, 2, ..., 10$, ואם יש למשל 15 אותיות ב־Γ הן הספרות $1, 2, ..., 14$, כאשר ה־_ הוא 0. כמו כן, אפשר לייצג את המצבים כמספרים, ואז ע"י א"ב של שלוש אותיות[21] אפשר להביע את כל העושר הזה - אפשר פשוט לייצג כל מספר מהמכונה המקורית כמספר בינארי ועוד אות שמבדילה בין המספרים. ישנן עוד הרבה דרכים לייצג זאת (מספרים מיוחדים ל־q_{acc}, q_{rej} למשל), הדרך עצמה לא מעניינת אותנו, אלא רק העובדה שאת כל הדברים הללו ניתן כאמור להציג כמספרים בינאריים ואות שתבדיל בין המספרים.

בהנתן מ"ט M, נסמן ב־$\langle M \rangle$ את הקידוד של M בא"ב של שלוש אותיות.

אם w הוא קלט למ"ט M, נסמן ב־$\langle w \rangle$ את הקידוד של w (יש צורך לקדד, שוב, מכיוון ואולי הא"ב של D קטן מזה של M. הקידוד נעשה באותה הדרך).

כעת שבידינו היכולת לקודד מכונת טיורינג, נוכל לדון בשאלות הכרעה של המכונה. נתחיל עם השאלה הבאה:

בהנתן מכונת טיורינג וקלט עבורה, האם מכונת הטיורינג עוצרת על הקלט?

נגדיר את הבעיה ע"י הגדרת השפה A_{TM}:

הגדרה 2.16 A_{TM} היא שפת כל המכונות והקלטים כך שהמכונה עוצרת על הקלט במצב מקבל, דהיינו:

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts w, meaning } w \in L(M)\}$$

[21] אפשר גם ע"י א"ב של שתי אותיות, ואפילו ע"י א"ב של אות אחת - אבל במקרה זה יש הבדל אקספוננציאלי באורך הקוד. כעת אנו לא מתעניינים ביעילות של האלגוריתמים שלנו, אבל זה יעניין אותנו בהמשך, לכן אנו נשאר עם הייצוג ע"י א"ב של שלוש אותיות.

נשאל:

1. האם $A_{TM} \in RE$?

תחילה, נבין משמעות השאלה הזו - האם קיימת מכונת טיורינג D שמקבלת קידוד של מכונת טיורינג אחרת וקלט, והיא עוצרת ומקבלת אמ"מ המכונה הנתונה עוצרת ומקבלת את הקלט הנתון?

הפתרון הנאיבי - "ש-D תריץ את M" - כאמור נחשוב על M כאלגוריתם, אז פשוט D תריץ את האלגוריתם M עם הקלט w, אם M יקבל יחזיר כן נחזיר כן, אם ידחה נחזיר לא, ואם יכנס ללופ גם אנחנו ניכנס ללופ, ואז הכל חוקי וטוב. לצורך דיון זה, נגדיר מכונת טיורינג מיוחדת:

2.1.1 מכונת טיורינג אוניברסלית

הגדרה 2.17 מכונת טיורינג U נקראת אוניברסלית, אם היא מקיימת את כל התנאים הבאים:

1. בהנתן קלט $\langle M, w\rangle$, אם M עוצרת בריצתה על w ומקבלת, אז U גם עוצרת ומקבלת.
2. בהנתן קלט $\langle M, w\rangle$ כך ש-M עוצרת על w ודוחה, אז U גם.
3. אם M אינה עוצרת בריצתה על w, אז בהנתן הקלט $\langle M, w\rangle$, U גם אינה עוצרת.
4. אם הקלט ל-U אינו קידוד חוקי של מכונת טיורינג וקלט עבורה, אז U עוצרת ודוחה.

נשאל, האם קיימת מכונת טיורינג אוניברסלית?

משפט 2.18 קיימת מכונת טיורינג אוניברסלית.

לא נוכיח את המשפט, כי הוא פשוט, ו"מצד שני, להוכיח אותו באופן פורמלי זה פשוט נורא" ארוך. יש כאן כל מיני שאלות תיכנותיות "בשפה העלובה של מכונת טיורינג", אולם המרצה בטוח שאנו יכולים לעמוד בהן.

האם המכונה האוניברסלית עוזרת לנו עם שאלה 1? כן!

נכניס את $\langle M, w\rangle$ למכונה האוניברסלית U , והיא תחזיר כן אמ"מ M מקבלת את הקלט - לכן השפה של U היא בדיוק ATM. יתרה מזו, אם M דוחה את w, מותר או לדחות או אף פעם לא לעצור - במקרה של מכונה אוניברסלית U, אם M דוחה, גם U תדחה. אם M לא עוצרת, גם U לא תעצור. אולם לפעמים אלו בדיוק המקרים שהכי מעניינים אותנו, מה שמביא אותנו לשאלה השניה:

2. האם $A_{TM} \in R$?

כלומר, האם קיימת מכונת טיורינג D המקבלת $\langle M, w\rangle$ כקלט, אשר מקבלת אם M מקבלת את הקלט w, D תקבל, ואם M לא מקבלת את w, D דוחה אותה.

התשובה לשאלה הזו - לא!

הרעיון: מה אנחנו דורשים מ-D? "זה קצת כמו לחזות את העתיד" - אם M עוצרת על w זה פשוט. אם M לא עוצרת, D צריכה לדעת ש-M לא תעצור על w. חיזוי העתיד אינו אפשרי: בהנתן אדם שחוזה את העתיד, אפשר לשאול אותו אם "נביא לו סטירה" עוד חמש דקות, ואפשר להשתמש במידע שיתן לנו כדי לסתור את ההנחה שלו על העתיד - אם יגיד שנסטור לו, לא נסטור לו, אם יגיד שלא נסטור לו - נסטור. דבר דומה נעשה כאן - נאמר שאם היתה קיימת כזו, אפשר לשאול אותה מה היא היתה עושה אם היינו נותנים לה את הקלט D. אם היא תגיד "אני אעצור", נתכנת אותה ככה שלא תעצור, ואם היא תגיד "אני אעצור", נבנה אותה בצורה שתכנס ללופ אינסופי. נעשה זאת בצורה פורמלית בשיעור הבא. 2/5/2012

משפט 2.19 $A_{TM} \notin R$.

הוכחה: נניח בשלילה כי $A_{TM} \in R$, כלומר קיימת מ"ט D המכריעה אותה. אזי, בהנתן קלט $\langle M, w\rangle$:

1. אם M עוצרת במצב מקבל בריצתה על w, D עוצרת במצב מקבל.
2. אם M אינה עוצרת, או עוצרת במצב דוחה בריצתה על w, D עוצרת במצב דוחה.

נבנה מ"ט D' כבציור: **

היא פועלת בצורה הבאה:

D' מקבלת קידוד של מכונת מ"ט M: $\langle M\rangle$, ומריצה את D על הקלט $\langle M, M\rangle$ - אם D מקבלת, D' דוחה, אם D דוחה, D' מקבלת.

נריץ את D' על $\langle D'\rangle$. D' עוצרת על $\langle D'\rangle$, שכן D תמיד עוצרת מהנחתנו. נבדוק את האפשרויות:

אם D' מקבלת, אזי D דוחה את $\langle D', D'\rangle$. מכאן (מהגדרת D) ש-D' אינה מקבלת בריצתה על $\langle D'\rangle$ - לא יכול להיות.

אזי, בהכרח D' דוחה בריצתה על $\langle D'\rangle$. מכאן נובע ש-D' עוצרת ומקבלת בריצתה על $\langle D'\rangle$, בסתירה. ■

סיכום ביניים:

$A_{TM} \notin R,\ A_{TM} \in RE$, ובפרט $R \neq RE$. זוהי הדוגמא הראשונה שראינו באופן מסודר, אבל יש עוד הרבה, אולם הן אינן בהכרח אינטואיטיביות (לופ של מכונות טיורינג כקלטים). מלבד אלו, יש בעיות שנראות לנו אלמנטריות - בעיות על מחרוזות למשל, שנראה לנו שיש אלגוריתם פשוט להכרעה של שפה שכזו. עם זאת, אין כזה אלגוריתם\אין כזו מכונת טיורינג.
עוד דוגמא שהזכרנו - שפת המשפטים שנובעים מסט האקסיומות של צרמלו (שזו הוכחה נוספת למשפט שגדל הוכיח).
נדון בהמשך גם בשפות שאינן ב-RE.

הגדרה 2.20 משפחת השפות $co-RE$ מוגדרת בצורה הבאה:

$$co-RE = \left\{ L \,\middle|\, \overline{L} \in RE \right\}$$

הגדרה 2.21 משפחת השפות $HALT$ מוגדרת באופן הבא:

$$HALT = \{\langle M, w\rangle \,|\, M \text{ halts on } w\}$$

הערה 2.22 זו משפחה שכן בתרגיל נגדיר את:

$$HALT^{\varepsilon} = \{\langle M\rangle \,|\, M \text{ halts on its run on } \varepsilon\}$$

ובשיטה דומה אפשר להגדיר את שאר השפות במשפחה.

טענה 2.23 $HALT \in RE$.

הוכחה: נשנה מכונה אוניברסלית כך שתקבל במקום לדחות. ■

טענה 2.24 $HALT \notin R$.

הוכחה: בצורה דומה להוכחה לגבי A_{TM}, רק שהפעם נגדיר את D' בצורה מעט שונה:

D' מקבלת קידוד של מכונת מ"ט M: $\langle M\rangle$, ומריצה את D על הקלט $\langle M, M\rangle$ -
אם D מקבלת, D' נכנסת ללופ, אם D דוחה, D' מקבלת.

והשאר - אותו הרעיון. ■

נרצה להוכיח את טענה זו בדרך אחרת: **הוכחה:** אנחנו יודעים שאין מכונה שמכריעה את A_{TM}. נניח בשלילה ש-$HALT \in R$, ונראה שמכך נוכל לבנות מכונה A שמכריעה את A_{TM}, בסתירה.
נבנה אותה בצורה הבאה:

תחילה, A מריצה את הקלט $\langle M, w\rangle$ במכונה H המכריעה את $HALT$.
אם $HALT$ דוחה את הקלט $\langle M, w\rangle$, אזי המכונה A תדחה.
אחרת, היא מקבלת, ואז נעביר אותה למכונה אוניברסלית U:
אם U מקבלת, נקבל.
אם U דוחה, נדחה.

נשים לב כי U לא יכולה להתקע, שכן וידאנו זאת ע"י המעבר המכונה H.
לכן A עוצרת תמיד, וכן מכריעה את A_{TM}, בסתירה למה שאנחנו כבר יודעים. ■

הערה 2.25 בהוכחה השניה הוכחנו ע"י רדוקציה לשאלה האם $A_{TM} \in R$.

ואפילו הוכחה שלישית, שתביא אותנו למקומות מעניינים - זוהי הוכחה ע"י רדוקציה של מיפוי.
תחילה, נוכיח טענת עזר:

טענה 2.26 קיימת מכונת טיורינג T אשר בהנתן קלט $\langle M, w\rangle$, היא מחשבת פלט: $\langle M', w\rangle$, כאשר M' מוגדרת באופן הבא:
אם M מקבלת את w, אזי M' מקבלת את w, אחרת M' אינה עוצרת לעולם בריצתה על w.

הוכחה: נחליף את המצב הדוחה ב-M במצב שמוביל ללולאה אינסופית לקבלת M'. ■

מסקנה 2.27 $HALT \notin R$.

הוכחה: נניח בשלילה $HALT \in R$, ותהא H מכונה המכריעה אותה. נבנה מכונה שמכריעה את A_{TM} באופן הבא:

בהנתן קלט $\langle M, w\rangle$, נריץ אותו תחילה ב־T לקבלת $\langle M', w\rangle$ מהטענה הקודמת. את הפלט הזה נריץ על H, ונחזיר את הפלט שלה.

מדוע מכונה זו מכריעה את A_{TM}?

אם M מקבלת בריצתה על w, אז בפרט M' תקבל בריצתה.

אם M לא מקבלת, M' נכנסת ללופ בריצתה על w, ולכן H תדחה את הפלט $\langle M', w\rangle$. ■

נשים לב כי T מקיימת:

$$x \in A_{TM} \Leftrightarrow T(x) \in HALT$$

זוהי שיטת הוכחה שחשוב להכיר. נגדיר אותה בצורה פורמלית:

2.2 רדוקציית מיפוי

הגדרה 2.28 יהיו L, L' שפות מעל א"ב Σ.
מ"ט T נקראת רדוקצית מיפוי מ־L ל־L' אם מתקיים:

$$\forall x \in \Sigma^* : \begin{array}{l} 1.\ T \text{ stops on } x \\ 2.\ x \in L \Leftrightarrow T(x) \in L' \end{array}$$

אם יש רדוקציית מיפוי מ־L ל־L', נסמן $L \leq_m L'$.

טענה 2.29 תהי T רדוקציית מיפוי מ־L ל־L'.
אזי, אם $L' \in R$, אזי בהכרח $L \in R$.

הוכחה: אם $L' \in R$, אז יש מכונת H המכריעה אותה. לכן המכונה הבאה מכריעה את L:

בהנתן קלט x, נריץ אותו תחילה על T, שתחזיר את הפלט $T(x)$. את הפלט הזה נריץ על H, ונחזיר את הפלט שלה.

אם $x \in L$, אזי $T(x) \in L'$ (שכן T היא רדוקציית מיפוי), ובמקרה זה H תקבל את הפלט.

אחרת $x \notin L$, אזי $T(x) \notin L'$, ולכן במקרה זה H תדחה את הפלט. ■

7/5/2012

נביט בשפה:

$$L = \{\langle M\rangle \,|\, L(M) \neq \phi\}$$

נרצה לבדוק באיזו מחלקות מאלו שאנחנו מכירים L נמצא או לא נמצאת, ובין השאר להשתמש ברדוקציה כדי לעשות זאת.

האם $L \in RE$?

כן! כיצד? אם נתחיל מהרצה של מילה כלשהיא במכונה M, אנחנו לא יודעים בכלל אם המכונה M תעצור!
נבנה מכונה D אשר "מריצה במקביל" את M על כל המילים ב־Σ^*, כאשר Σ היא א"ב הקלט של M.
אם M לא עוצרת על אף מילה, אין לנו בעיה עם כך ש־D גם לא תעצור, שכן אנו מחפשים מכונה שמזהה את L, לא מכריעה אותה.
כיצד רצים במקביל?
יש כמה דרכים לעשות כך. אנו נעבור על $i = 0, 1, 2, ...$, ולכל i, נריץ את M במשך i צעדים על כל אחת מהמילים ב־Σ^* באורך $\leq i$. אם M עוצרת במצב מקבל באחת ההרצות הללו, נחזיר "כן".
למה נריץ על מילים באורך $\leq i$ ולא i? יכול להיות ש־M מגיעה למצב מקבל על מילה לאחר מספר צעדים גדול מאורכה, ואז, אם נוותר על ריצה על יתר המילים, לא נגלה כי M מקבלת בריצה עליה.
אזי, בבירור D מזהה את השפה L, ולכן $L \in RE$.

האם $L \in R$?

טענה 2.30 $L \notin R$.

הוכחה: כדי להראות זאת, נמצא רדוקציית מיפוי ל־L מבעיה שאנחנו יודעים שאינה ב־R. נשתמש ב־A_{TM}.
נבנה מכונת טיורינג T, אשר בהנתן קידוד $\langle M, w\rangle$ חוקי, מחזירה קידוד $\langle M'\rangle$ כדלהלן:
בהנתן קלט X, M' תסמלץ את ריצת M על הקלט w (ללא קשר לקלט שלה X).
אם הסימולציה עוצרת במצב מקבל, M' גם תעצור במצב מקבל. אם הסימולציה עוצרת במצב דוחה, M' תדחה.
אם M מקבלת את w, כלומר $\langle M, w\rangle \in A_{TM}$, אזי $L(M') = \Sigma^*$, ולכן $\langle M'\rangle \in L$. מצד שני, אם $\langle M, w\rangle \notin A_{TM}$, אזי $L(M') = \phi$, ולכן $\langle M'\rangle \notin L$.
על כן, T היא רדוקציית מיפוי.

הערה 2.31 נשים לב כי ע"מ ש־T תהיה רדוקציית מיפוי, במידה והקלט אינו חוקי, על T להחזיר פלט שלא יהיה ב־L: לא קידוד חוקי של מכונת טיורינג, למשל, או לבחור מכונה שלא ב־L. לא נזכיר זאת בהמשך, אבל באופן פורמלי יש לכתוב זאת במבחן\בוחן.

לפי הטענה בסוף השיעור הקודם, אם $L' \in R$, ו־$L \leq_m L'$, אזי $L \in R$. לכן אם נניח כי $L \in R$, מכיוון וראינו כי $A_{TM} \notin R$, L לא יכולה להיות ב־R, בסתירה. ■

כעת, נביט בשפה:

$$L_1 = \{\langle M\rangle \,|\, L(M) \neq \Sigma^*,\ L(M) \neq \phi\}$$

האם $L_1 \in R$?

לא!

האם $L_1 \in RE$?

לא!

האם $L_1 \in co-RE$?

לא!
אבל... מסתבר שלא נראה את זה :_(זה כי זה בתרגיל 8!

$$L = \{\langle M\rangle \,|\, |L(M)| = \infty\}$$

טענה 2.32 $A_{TM} \leq_M L$.

הוכחה: בהנתן קלט $\langle M, w\rangle$ לרדוקציה, הרדוקציה תחזיר קידוד $\langle M'\rangle$ של מ"ט כדלהלן:
בהנתן קלט x עבור M', M' תסמלץ על ריצתה של M על w. אם M מקבלת את w, M' תקבל את x. אחרת, אם M עוצרת על w, M' תדחה. נראה שזוהי אכן רדוקציה:

- הרדוקציה שהגדרנו תמיד עוצרת.
- אם $\langle M, w\rangle \in A_{TM}$, אז $L(M') = \Sigma^*$, ולכן $\langle M'\rangle \in L$.
- אם $\langle M, w\rangle \notin A_{TM}$, אזי $L(M') = \phi$, ולכן $\langle M'\rangle \notin L$.

■

טענה 2.33 $(A_{TM})^c \leq_M L$.

הוכחה: תחילה, עלינו להבין מיהי $(A_{TM})^c$. מהקלטים שהם אינם קידודים חוקיים של M, w נתעלם. נתייחס רק לקלטים שהם קידודים חוקיים של M, w, והם בשפה הזו אם M אינה מקבלת את w.

בהנתן $\langle M, w\rangle$, הרדוקציה תחזיר קידוד $\langle M'\rangle$ של מ"ט כדלהלן:

בהנתן קלט x ל־M', M' תסמלץ את ריצת M על w במשך $|x|$ צעדים. אם M קיבלה את w במהלך הסימולציה, M' תדחה. אחרת, M' תקבל.

- הרדוקציה תמיד עוצרת, שכן ניתן לכתוב את הקידוד של M' בזמן סופי.
- אם $\langle M, w\rangle \in (A_{TM})^c$, אזי $L(M') = \Sigma^*$, ובפרט $|L(M')| = \infty$, ועל כן $\langle M'\rangle \in L$.
- אם $\langle M, w\rangle \notin (A_{TM})^c$, $\langle M, w\rangle \in A_{TM}$, כלומר M מקבלת את w, אזי $L(M')$ מכילה רק מילים שאורכן קצר מאורך הריצה של M על w, ולכן $|L(M')| < \infty$, ולכן $\langle M'\rangle \notin L$.[22]

■

קיבלנו אם כך כי $A_{TM} \leq_M L$, וגם $(A_{TM})^c \leq_M L$.

מסקנה 2.34 $L \notin R$.

תזכורת:

הראינו כי אם $L_1 \leq_M L_2$:

אם $L_2 \in R$־, אז $L_1 \in R$.

בעצם, בהנתן התנאי מעלה, ניתן להראות:

- אם $L_2 \in RE$, אז $L_1 \in RE$.
- אם $L_2 \in co-RE$, אז $L_1 \in co-RE$.

מסקנה 2.35 $L \notin co-RE$.

הוכחה: $A_{TM} \notin co-RE$[23], ו־$A_{TM} \leq_M L$, ולכן $L \notin co-RE$. ■

מסקנה 2.36 $L \notin RE$.

הוכחה: $(A_{TM})^c \notin RE$ [24], והראינו $(A_{TM})^c \leq L$, לכן $L \notin RE$. ■

2.2.1 רדוקציה בשפות שאינן מתוארות כמכונות טיורינג

למשל[25]:

הגדרה 2.37 השפה PCP מוגדרת באופן הבא:

הקלט: א"ב Σ, ורשימה $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, ..., \begin{pmatrix} x_n \\ y_n \end{pmatrix}$ של זוגות של מילים לא ריקות מעל Σ.

הפלט: "כן" אמ"מ קיימת סדרה סופית של אינדקסים באורך כלשהוא (עם חזרות) $i_1, i_2, ..., i_t$ כך ש:

$$x_{i_t}, x_{i_2}, ..., x_{i_t} = y_{i_1}, ..., y_{i_t}$$

---

[22] זה נראה מסובך, אבל גיא בטוח ש"אחרי חמש דקות לבד בחושך... או לא בחושך" נבין "מה המכונה פולטת". אני רצינית, זה ציטוט (אולי קצת ערוך, but you get the jist).

[23] כי איננה ב־R אבל כן ב־RE, משאלה 2 תרגיל 6 אם היתה גם ב־$co-RE$ היתה ב־R, בסתירה.

[24] כי אם $(A_{TM})^c \in RE$, אזי מההגדרה זה אומר ש־$A_{TM} \in co-RE$, אבל זה כאמור לא יכול להיות (ראה הערה קודמת).

[25] PCP הוא ראשי תיבות של כל מיני דברים, למשל של חומר בשם angel dust, שגורם לדברים לא מהנים במיוחד.. נקווה שהכוונה שלנו כאן מהנה יותר.

דוגמא: הקלט: $\Sigma = \{a, b\}$, והרשימה היא:

$$\left(\begin{array}{c} a \\ ab \end{array} \right), \left(\begin{array}{c} b \\ a \end{array} \right), \left(\begin{array}{c} abab \\ ab \end{array} \right)$$

הפתרון: "כן": $1, 2, 1, 3$, שכן נקבל:

$$\begin{array}{cccc} 1 & 2 & 1 & 3 \\ a & b & a & abab \\ ab & a & ab & ab \end{array}$$

הגדרה 2.38 השפה $M - PCP$ היא:
קלט: כמו קודם, אלא שנתון גם זוג מיוחד $\left(\begin{array}{c} x_0 \\ y_0 \end{array} \right)$, שבו ניתן וגם חייבים להשתמש בתחילת השרשור בלבד.
פלט: "כן" אמ"מ קיימת סדרה סופית (יכולה להיות ריקה) של אינדקסים $i_1, i_2, ..., i_t$ כך כך ש־

$$x_0, x_{i_1}, x_{i_2}, ..., x_{i_t} = y_0, y_{i_1}, y_{i_2}, ..., y_{i_t}$$

טענה 2.39 $M - PCP \leq_M PCP$.

הוכחה: נתאר את הרדוקציה באופן לא לחלוטין פורמלי:
בהנתן Σ, $\left(\begin{array}{c} x_n \\ y_n \end{array} \right), ..., \left(\begin{array}{c} x_1 \\ y_1 \end{array} \right), \left(\begin{array}{c} x_0 \\ y_0 \end{array} \right)$, הרדוקציה תפעל כדלהלן:
הרדוקציה תחזיר את הא"ב $\Sigma \bigcup \{*, !\}$. את המחרוזות הרדוקציה תבנה כך:
עבור מחרוזת $w = w_1, ..., w_m$, $w \in \Sigma^*$, נגדיר:

$$"*w" = *w_1 * w_2 .... * w_n$$
$$"w*" = w_1 * w_2 * ...w_m *$$
$$"*w*" = *w_1 * w_2 * ... * w_m *$$

הרדוקציה תחזיר את המחרוזות:

$$\left(\begin{array}{c} *x_0* \\ *y_0 \end{array} \right), \left(\begin{array}{c} x_1* \\ *y_1 \end{array} \right), \left(\begin{array}{c} x_2* \\ *y_2 \end{array} \right), ..., \left(\begin{array}{c} x_n* \\ *y_n \end{array} \right), \left(\begin{array}{c} ! \\ *! \end{array} \right)$$

נשתכנע שהרדוקציה עובדת:
תחילה, ברור כי היא עוצרת.
שנית, אם הקלט לרדוקציה הוא בשפה $M-PCP$, אזי קיימת סדרת אינדקסים $i_1, ..., i_t$ שפותרת אותו. הסדרה שפותרת את הפלט (ב־PCP) של הרדוקציה הינה:

$$0, i_1, ..., i_t, (n+1)$$

כאשר האינדקס האחרון מתייחס לזוג הימני ביותר שהוספנו.
ולסיום, נראה שאם הפלט ב־PCP, אזי הקלט ב־$M - PCP$:
נביט בשרשור של הפלט מ־PCP. אם נתחיל בכל אחד מהזוגות האחרים ולא הראשון, בשל הכוכביות, לא נוכל לקבל שיוויון - הוא היחיד שמאפשר כוכבית במקום הראשון גם למעלה וגם למטה. לאחר מכן אפשר להשתמש בכל אחד מהזוגות האחרים, והכוכביות יסתדרו, ולא ניתן להשתמש בזוג המיוחד $\left(\begin{array}{c} *x_0* \\ *y_0 \end{array} \right)$ עוד פעם, או בזוג המיוחד האחרון, ובסוף השרשור הזוג האחרון משלים לשרשור חוקי.

כלומר, הדרך היחידה להרכבת אותה המילה למעלה ולמטה היא להתחיל עם הזוג $\left(\begin{array}{c} *x_0* \\ *y_0 \end{array} \right)$, להמשיך עם הזוגות האחרים שאינם האחרון, ולסיים עם האחרון.
על כן, נמחוק מסדרת האינדקסים את האינדקס הראשון והאחרון, ונקבל סדרה שמהווה פתרון ל־$M - PCP$, כנדרש.■

טענה 2.40 $A_{TM} \leq_M M - PCP$.

הוכחה: בהנתן $\langle M, w \rangle$, הרעיון:
הזוג המיוחד יהיה בנוי באופן הבא:

$$\begin{pmatrix} \#\,(q_0, w_1)\,w_2, ..., w_{|w|}\# \\ \# \end{pmatrix}$$

והזוגות יאפשרו להעתיק למטה מה שכבר כתוב למטה (נטפל באקסטרה סולמיות בסוף):

$$\begin{pmatrix} \# \\ \# \end{pmatrix}, \left\{ \begin{pmatrix} \alpha \\ \alpha \end{pmatrix} \right\}_{\alpha \in \Gamma}$$

לא ניתן דרך להעתיק למטה את הזוג (q_0, w_1), אלא רק כשרשום למעלה המצב העוקב - כלומר, אם $\delta(q, \alpha) = (q', \beta, R)$, נוסיף את:

$$\begin{pmatrix} (\beta\,(q', \gamma)) \\ (q, \alpha)\,\gamma \end{pmatrix}$$

לכל אות אפשרית $\gamma \in \Gamma$. אז כל פעם שנעתיק נקבל את הקונפיגורציה הבאה למעלה, וכך נקבל את השלבים של הריצה של המכונה. נראה את זה ואיך מסדרים את הבעיה של הכוכביות בהמשך... מסתבר שלא הראנו את זה, חבל. ■

/5/2012+14/5/2012

לסיכום PCP:

$$(M - PCP \leq_m PCP) + (A_{TM} \leq_m M - PCP)$$
$$\Rightarrow \overbrace{A_{TM}}^{\notin R} \leq_m \overbrace{PCP}^{\notin R}$$

מצד שני, $PCP \in RE$, כי אם אוסף הזוגות בשפה, בעזרת מכונה המנסה את כל האפשרויות ניתן למצוא את המחרוזת (לו היה לנו עוד זמן היינו רואים עוד שפות כאלו).
יש שפות על ריצוף במישור - אוסף צורות (טטריס), האם ניתן לרצף בעזרתן את המישור כולו? או למשל, בהנתן מערכת משוואות פולינומיות, האם למערכת יש פתרון במספרים טבעיים? הבעיה ב-RE, ויש רדוקציה מ-A_{TM} אליה.
ראינו בעיה נוספת - שפת המשפטים המתמטיים הניתנים להוכחה מ-ZFC, שגם היא ב-$RE \backslash R$ - בהנתן מחרוזת בא"ב ההוכחות, ניתן לוודא האם היא הוכחה חוקית. שפה זו ניתנת לזיהוי ע"י מעבר על כל המחרוזות. מצד שני, יש לה רדוקציה מ-A_{TM}.
נשאל - האם כל שפה $L \notin co - RE$ מקיימת $A_{TM} \leq_m L$?

הגדרה 2.41 נאמר ששפה L היא RE-קשה, אם לכל שפה $L' \in RE$, מתקיים $L' \leq_m L$.

טענה 2.42 אם שפה L מקיימת $A_{TM} \leq_m L$, אזי L היא RE-קשה.

הוכחה: מתרגיל 8, לכל $L' \in RE$ מתקיים $L' \leq_m A_{TM}$. מטרנזיטיביות היחס, נקבל כי $L' \leq L$. ■

הגדרה 2.43 שפה נקראת RE-שלמה אם $L \in RE$, וגם L היא RE-קשה.

עד כה, לגבי שפות שהן לא ב-R, ראינו רק RE-קשות או RE-שלמות.

טענה 2.44 אם L_1, L_2 הן RE שלמות, אזי $L_1 \leq_m L_2$.

יש שפות ב-RE שאינן RE-שלמות, אך זה לא פשוט ליצרן. הבעיות הטבעיות הן משום מה RE-שלמות אם הן ב-$RE \backslash R$. צריך לעבוד קשה כדי ליצר בעיות כאלו, וצריך לעבוד קשה כדי להראות שאין לא שלמות (אין רדוקציה מ-A_{TM} אליה), ולא כריעות.
ראינו שפות מחוץ ל-$RE \cup co - RE$. בתוך $co - RE$ אפשר לדבר על שפות $co - RE$-שלמות, ואם היינו רוצים, היינו יכולים להגדיר מחלקות מעל $co - RE$, RE, ועדיין למצוא שפות שהן לא בהן.
אבל, כל הדברים הללו הם תיאורטיים, ואנחנו לא באמת מאמינים שניתן לעשות משהו מעבר ל-RE, כי TM נחשב למודל החישובי המציאותי הכי חזק שקיים.
לגבי שפת המשפטים הניתנים להוכחה ב-ZFC, ציינו כי היא נמצאת ב-$RE \backslash R$, וכי יש אליה רדוקציה מ-A_{TM}.

מסקנה 2.45 יש משפט מתמטי נכון שאינו ניתן להוכחה.

הוכחה: בהנתן $\langle M, w\rangle$, נביט במשפט "M איננה עוצרת על w". אם המשפט לא נכון, אזי אפשר להפריך אותו ולהוכיח שאינו נכון. אם תמיד כשהמשפט נכון אפשר להוכיח את זה, אזי ניתן להכריע את A_{TM}, בסתירה להוכחה שלנו משיעור קודם שהיא בלתי ניתנת להכרעה. על כן, יש משפט מהסוג הזה שהוא נכון, ולא ניתן להוכיח אותו. ■

הערה 2.46 זהו חלק ממשפט אי השלמות של גדל.

גדל הוכיח כי בכל מערכת אקסיומות עקבית "עשירה מספיק", ישנם משפטים שאינם ניתנים להוכחה. אנחנו הראינו על ZFC.

טיורינג המציא במקור את מכונת טיורינג על מנת להוכיח את המשפט הנ"ל בצורה דומה למה שאנו עשינו עד כה (אבל בצורה כללית יותר):

תהא M מ"ט, ותהא M' מ"ט המדפיסה על הסרט שלה את הקידוד $\langle M'\rangle$, ולאחר מכן רצה כמו M[26]. נתכנת את M כך שבתחילת ריצתה היא כותבת את המשפט "$\langle M\rangle$ איננה עוצרת על הקלט הריק", ולאחר מכן עוברת על כל ההוכחות האפשריות למשפט הזה, ועוצרת אם היא מצאה הוכחה.

משפט 2.47 בהנתן M כנ"ל, המשפט "$\langle M\rangle$ איננה עוצרת על הקלט הריק" נכון, אבל לא ניתן להוכחה במערכת ZFC.

כלומר, יש משפט ספציפי שאיננו ניתן להוכחה. זה נכון לכל מערכת היסק ש"מספיק עשירה" כדי לנסח אתה המשפט הזה.

טיורינג כאמור התעניין במכונות טיורינג על מנת לתת כלי להוכחת משפט אי השלמות של גדל, ופחות בשאלה האם המודל של מכונת טיורינג מספיק חזק כדי לסמלץ כל מחשב קיים או עתידי.

מודל המחשב אותו אנו מכירים היום נקרא מודל $Von-Neumann$:

ניקח את היקום, ונחלק אותו לקוביות מאוד קטנות. כדי לדמות את מצב היקום בזמן מסויים, נכתוב את המצב $f(t,x)$ בכל משבצת x של היקום, ואת כל זה נזין לזיכרון של ה"מחשב". על מנת לחשב את המצב הבא של היום, CPU יעבור על כל אחד מהתאים בזיכרון, ומ־$f(tx)$ והתאים שליד יחשב את מצב היקום בזמן $t+\Delta t$.

המחשב הראשון היה עם מפסקים אנלוגיים. למרות שהטכנולוגיה השתנתה, רב המחשבים עדיין עובדים לפי המודל הזה, שמתאים לפתרון משוואות דיפרנציאליות. בשלב מסויים הוטלו מגבלות על הכוח של מחשבים חזקים.

בעזרת המודל הזה ניתן לדמות את חוקי הפיזיקה הקלאסית. אם רוצים להוסיף למשל את חוקי התורה היחסותית, יש רק להגדיר את המעבר מ־$f(t,x)$ ל־$f(t+\Delta t,x)$ בצורה שונה. המודל הזה הוא מודל של מכונת RAM אותו ראינו בתרגול, אשר ניתן לדימוי בעזרת מ"ט.

נרצה לטעון כי לא ניתן לבנות מכונה חזקה יותר ממכונת טיורינג, כלומר, שתוכל להכריע שפות שמ"ט איננה יכולה להכריע. אם נבנה מכונה כזו מדברים הקיימים בעולם שלנו, הוא יהיה בנוי מאלמנטים פיסיקליים. נוכל לכתוב מה יש במחשב בכל נקודה, ואז לסמלץ את הפעולה הפיזיקלית של המכונה הזו באותה צורה שפון־ניומן סמלץ פיצוצים אטומים, למשל, ואם ניתן לעשות דבר בעזרת מכונת RAM, אזי בוודאי שאפשר לעשות את אותו הדבר עם מ"ט.

כל עוד הפיזיקה עונה על התנאי שניתן לחשב את $f(t+\Delta t,x)$ מתוך $f(t,x)$ בסביבה הקרוב של x, נוכל לסמלץ את המכונה.

התזה של צ'רץ' טיורינג

טענה 2.48 אוסף השפות הניתנות להכרעה\זיהוי ע"י מחשב כלשהו, הוא RE או R בהתאמה.

אפשר לשאול, איך נדע האם החלוקה שלנו מספיק מדוייקת, או אולי האנרגיה בנקודה מסויימת תחרוג ממספר הביטים הדרושים לביטוייה.

[26] חלק מהתרגולים ראו את זה.

3 משפחות מוגבלות משאבים

עד כה ראינו את אוסף השפות הכריעות, מעליו יש את RE ו־$co-RE$, ומעליהן שפות שהן לא ב־RE ולא ב־$co-RE$.
מה לגבי השפות הקלות ביותר?
עליהן נדבר ביתר הקורס.
כבר ראינו כי $REG \subsetneq R$. גם עבור השפות הכריעות, ניתן לדון בכמות המשאבים הדרושה להכרעת שפה.
המשאב הראשון שנדבר עליו הוא:

3.1 משאב הזמן

הגדרה 3.1 תהא $L \in R$, ותהא M מ"ט המכריעה את L. נגדיר:

$$t_M(n) = \max_{|x| \leq n} \{\#\text{ of steps in the run } M(x)\}$$

נגיד ש־M רצה בזמן $O(f(n))$, אם $t_M(n) \in O(f(n))$. באופן דומה ניתן להגדיר גם עבור Ω ו־Θ.

הערה 3.2 נשים ♡ כי הזמן הוא תכונה של המכונה, לא של השפה!

נגיד ש־$L \in TIME[f(n)]$, אם יש מ"ט M המכריעה את L, הרצה בזמן $O(f(n))$.

דוגמא

נביט בשפה:

$$L = \left\{ w \in \{a,b\}^* \middle| \#_a(w) = \#_b(w) \right\}$$

האם $L \in TIME\left[n^2\right]$? כן!
עם שני סרטים אפשר בזמן לינארי, ועם סרט אחד לא ידוע.

עכשיו אנו רואים שהמחלקה $TIME[n]$ תלויה במודל של מ"ט בו משתמשים. כעת נדבר תמיד על המודל בעל סרט בודד.

תרגיל: הוכיחו כי $L \in TIME[n \log n]$.

היינו רוצים לחפש מחלקות ששינוי במודל לא ישנה, וזו המחלקה הבאה:

הגדרה 3.3

$$P = \left\{ L \in \Sigma^* \middle| \Sigma \text{ set}, \exists k,\ L \in TIME\left[n^k\right] \right\} = \bigcup_{k=1}^{\infty} TIME\left[n^k\right]$$

אלו כל השפות שניתן להכריע בזמן פולינומי.

23/5/20122

לפי התרגול, מחלקה הזו לא משתנה באם משתמשים במודל RAM, או מודל מ"ט עם שני סרטים.
נשים לב כי $REG \subseteq TIME[n]$.
כמו כן:

$$P \subseteq TIME[2^n]$$

הגדרה 3.4

$$EXP_{TIME} = \bigcup_{k=1}^{\infty} TIME\left[2^{n^k}\right]$$

הינו אוסף כל השפות שניתן להכריע אותן בזמן אקספוננציאלי.

בעיות ומחלקותיהן

- קשירות בגרף - פולינומי, כלומר ב־P.
- קיום זיווג מושלם - פולינומי, כלומר ב־P.
- פתירות משוואות לינאריות - ב־P.
- A_{TM} - אין מכונה שמכריעה את השפה, ולכן בפרט לא קיימת מכונה שמכריעה אותה בזמן פולינומי או אקספוננציאלי, לכן $A_{TM} \notin P, EXP, R$.
- ראשוניות: נתון מספר בקידוד בינארי, יש להכריע אם המספר ראשוני או לא - מספר הביטים בקלט הוא $\log n$. אם נעבור על כל המספרים הקטנים מ־$\sqrt{n}$ ונבדוק אם המספר מחלק אותם, זה יפתור את הבעיה, אבל זה לא פולינומי. שאלת השיוך היתה פתוחה עד 2004, אז מצאו אלגוריתם פולינומיאלי לפתרון הבעיה, כלומר הבעיה ב־P.
- $SUBSET - SUM$: בהנתן מספרים טבעיים $\{a_1, a_2, ..., a_n\}$, ומספר נוסף t, יש להכריע אם ניתן להציג את t כסכום של איברים שונים מתוך $\{a_1, a_2, ..., a_n\}$. לא ידוע אם בעיה זו ב־P (ההשערה היא שלא). בבירור בעיה זו ב־EXP_{TIME}.
- SAT: ב־EXP_{TIME}, לא ידוע אם היא ב־P (ההשערה היא שלא).
- בעיה: בהנתן קידוד של מ"ט $\langle M \rangle$ וקידוד של קלט $\langle w \rangle$, האם M מקבלת את w בזמן $\leq 2^{\left(2^{|w|}\right)}$. בעיה זו **אינה** ב־$P$. בשביל לדעת באיזו מחלקת זמן בעיה זו, צריך לעבור שוב על הבניה של מכונה אוניברסלית, ולראות כמה צעדים יקח למכונה המכריעה לסמלץ את M. בעיה זו כן ב־R.
- בעיית $HAM - CYCLE$ היא הבעיה הבאה:
  בהנתן גרף, יש להכריע אם יש בו מעגל העובר בכל צומת פעם אחת בדיוק.
  אנו לא יודעים כיצד להכריע אותה בזמן פולינומי.
- בעיית $INDEPENDENT - SET$, או בקיצור IS, היא הבעיה הבאה:
  בהנתן גרף G ומספר k, יש להכריע אם יש קב' צמתים בגודל k ב־G כך שאין קשתות הצמתים בקבוצה.
  לא יודעים לרדת מזמן אקספוננציאלי בעת פתרון בעיה זו, וכמו כן לא יודעים שהיא אינה ב־P.

נדבר על בעיות נוספות בהמשך.
לכל הבעיות שדיברנו עליהן והן ב־EXP_{TIME} ולא ידוע אם הן ב־P יש דבר מה משותף:
אנו לא יודעים לפתור את הבעיה עצמה, אבל אם נתון לנו פתרון ספציפי, אפשר לוודא בזמן פולינומי שזהו אכן פתרון.
זוהי תכונה מעניינת שנדבר עליה רבות.
איזו בעיה אותה ראינו אינה בעלת התכונה הזו (או לא ברור שהיא בעלת התכונה)?
נביט ב־$\overline{HAM - CYCLE}$: בהנתן גרף, יש להכריע אם אין מעגל בגרף שעובר על כל צומת פעם אחת ויחידה. לבעיה זו אין התכונה הנ"ל.

3.2 מכונות טיורינג לא דטרמניסטיות

ראינו בתרגול את ההגדרה למכונת טיורינג אי־דטרמיניסטית (NTM):

הגדרה 3.5 מכונת טיורינג אי דטרמיניסטית (NTM) זהה למ"ט רגילה (TM), מלבד שינוי בפונקציית המעברים שלה:

$$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{R,L\}}$$

כלומר, ריצה של NTM על מילה w היא סדרת קונפיגורציות (כמו ב־TM) כך שכל קונפיגורציה עוקבת לזו שלפניה לפי δ.
נאמר ש־N NTM מקבלת מילה w אם קיימת ריצה מקבלת של N על w.
השפה של N NTM היא:

$$L(N) = \{w | N \text{accepts } w\}$$

ונאמר ש־N מזהה את $L(N)$.
בהתבסס על הגדרה זו, נגדיר את המושג הבא:

הגדרה 3.6 בהנתן מ"ט לא דטרמיניסטית M,

$$t_M[n] = \max_{|w|\leq n}\{\text{Longest run time of } M \text{over } w\}$$

ריצות ש"נתקעות" נחשוב עליהן כריצות שעוצרות ונדחות.
נאמר שמכונת טיורינג לא דטרמיניסטית M רצה בזמן $O(f(n))$, אם $t_M[n] \in O(f(n))$.
עבור $f : \mathbb{N} \to \mathbb{N}$, נגדיר:

$$NTIME[f(n)] := \{L \text{ is a language over } \Sigma | \exists \text{a non-deter' TM s.t. recognizes } L \text{in time } O(f(n))\}$$

$$NP := \bigcup_{k=1}^{\infty} NTIME(n^k)$$

$$NEXP_{TIME} = \bigcup_{k=1}^{\infty} NTIME\left[e^{n^k}\right]$$

נשים לב:

$$TIME[f(n)] \subseteq NTIME[f(n)]$$
$$\Rightarrow P \subseteq NP$$

הלוגיקאי הוק הגדיר את המחלקות, ושאל האם יש שיוויון כבר בשנת '72, אולם עד היום אין תשובה.

טענה 3.7

$$NP \subseteq R$$

הוכחה: אם $L \in NP$ מעל Σ^*, אז יש מ"ט ל"ד M, $k \in \mathbb{N}$ וגם (בה"כ) $C \in \mathbb{N}$, כך ש:

1. M מזהה את L.
2. על קלט w, כל ריצה של M על w עוצרת בזמן $\leq C \cdot n^k$, עבור $|w| = n$.

ניתן להכריע את L ע"י מ"ט דטרמיניסטית D אשר בהנתן w, תסמלץ את כל הריצות האפשריות של M על w, ותחזיר "כן" אמ"מ אחת מהריצות מקבלת. ■

$NP \subseteq EXP$, יכול להיות שהוא שווה לו, וכמו כן כאמור יכול להיות ש־$NP = P$. זהו מצב הידע הנוכחי בעולם.
מה שנראה באחד מהשיעורים הבאים, הוא כי $P \subsetneq EXP$.

טענה 3.8 $NP \ni HAM - CYCLE$.

הוכחה: נבנה מ"ט ל"ד שמזהה את $HAM - CYCLE$ בזמן פולינומי באופן הבא:
בהנתן גרף G בעל n צמתים, המכונה M תכתוב באופן לא דטרמיניסטי סדרה באורך n של צמתים מהגרף. אח"כ נבדוק אם הסדרה הנ"ל היא מעגל המילטוני, ונקבל אם כן.
M רצה בזמן פולינומי, ואכן מזהה את $HAM - CYCLE$. ■

הערה 3.9 שיטת הוכחה זו ניתן להפעיל עבור כל שפה בעלת התכונה שציינו בתחילת השיעור. כך למשל ניתן להוכיח גם עבור SAT, דהיינו $SAT \in NP$. באותו אופן, גם $SUBSETSUM - SUM \in NP$.

ל־$\overline{HAM - CYCLE}$ אין את התכונה הזו, על כן הוכחה שכזו עבור השפה לא תעבוד.
ישנה מחלקה נוספת שנקראת $co - NP$. $\overline{HAM - CYCLE} \in co - NP$. 28/5/2012
ראינו את המושג של רדוקציית מיפוי כאשר דיברנו על R ו־RE. נראה את ההרחבה למושג לנושא הנוכחי:

3.3 רדוקציית מיפוי פולינומית

3.3.1 הגדרות

הגדרה 3.10 רדוקציית מיפוי פולינומי היא רדוקציית מיפוי, המסיימת את ריצתה הזמן פולינומי.
אם יש רדוקציית מיפוי פולינומית משפה L_1 לשפה L_2, נסמן $L_1 \leq_p L_2$.

כל הרדוקציות שראינו בעבר היו למעשה רדוקציות מיפוי פולינומיות.

הערה 3.11 כאשר מדברים על רדוקציית מיפוי, נשים לב כי המכונה היא מכונה **דטרמיניסטית**.

טענה 3.12 אם $L_1 \leq_p L_2$ ו־$L_2 \in P$, אזי $L_1 \in P$.

הוכחה: בהנתן אלגוריתם פולינומי המכריע את L_2, נוכל להרכיב אותו עם הרדוקציה, ולקבל אלגוריתם המכריע את L_1. ■

טענה 3.13 אם $L_1 \leq_p L_2$ ו־$L_2 \in NP$, אזי $L_1 \in NP$.

הוכחה: בהנתן מכונת טיורינג לא דטרמיניסטית הפועלת בזמן פולינומי המזהה את L_2, נוכל להרכיב אותה עם הרדוקציה, ולקבל אלגוריתם המזהה את L_1. ■

מסקנה 3.14 אם $L_1 \leq_p L_2$ ו־$L_1 \notin P$, אזי גם $L_2 \notin P$.

3.3.2 דוגמאות

נזכר בבעית ה־$CNF - SAT$, המקבלת ארגומנטים מהסוג הבא:

$$(x_1 \vee \overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_4 \vee x_3 \vee x_{17}) \wedge ...$$

Independent Set - IS היא הבעיה הבאה:
בהנתן (G, k) כאשר G הוא גרף לא מכוון ו־$k \in \mathbb{N}$, יש להחזיר "כן" אמ"מ יש ב־G קבוצת צמתים בגודל k לפחות שאין ביניהם קשתות.

ראינו (בערך) רדוקציה מ־IS ל־$CNF - SAT$, נראה בכיוון ההפוך:

טענה 3.15 $CNT - SAT \leq_p IS$.

הוכחה: נבנה רדוקציה R כדלהלן:
בהנתן נוסחת $CNF - SAT$ φ, R תבנה גרף בו יש צומת לכל הופעה של ליטרל ב־φ, וכן תבנה קשת בין שני מופעים של ליטרל אם הם מופיעים באותו הסגר, וכן תבנה קשת בין שני קודקודים אם הם סותרים. k יהיה מספר ההסגרים.

למשל, עבור הנוסחא $(x_1 \vee \overline{x}_2) \wedge (x_2 \vee x_1 \vee x_3)$ הרדוקציה תיתן גרף הנראה כך: *ציור* כאשר נקבל מעגל בין x_3 , $x_2, (x_1, 2)$, $(x_1, 1)$ מקושר ל־$\overline{x}_2$ המחובר ל־x_2, ו־$k = 2$. במקרה זה ההשמות $\mathbb{T}, \mathbb{T}, \mathbb{T}$ יספקו את הנוסחא.

נכונות: 1. הרדוקציה רצה בזמן פולינומי.
2. נוכיח כי אם $\varphi \in CNF - SAT$, אז $R(\varphi) \in IS$:
תהא A השמה מספקת עבור φ, ונניח כי ב־φ יש k הסגרים. נבחר מכל הסגרים ב־φ ליטרל אחד שערך האמת ש־A מקנה לו הוא $TRUE$. נסתכל בקבוצת הצמתים המתאימים למופעי הליטרלים שבחרנו. אלה מהווים IS בגודל הרצוי. מכיוון שבחרנו מופע יחיד מכל הסגר, אין בין הצמתים קשתות מסוג 1 (אלה שבין מופעים מאותו הסגר). מכיוון שערך כל הליטרלים הוא $TRUE$ אין ביניהם קשתות מסוג 2.
3. נוכיח כי אם $R(\varphi) \in IS$ אזי $\varphi \in CNF - SAT$:
יהא k מספר ההסגרים ב־φ, ותהא S קבוצה בלתי תלויה בגודל k ב־$R(\varphi)$.
אבחנה: מופעי הליטרלים המתאימים לצמתי S מופיעים כולם בהסגרים שונים, אחרת היתה ביניהם קשת, בסתירה לכך ש־S בלתי תלויה. כמו כן, יש השמה למשתני φ המקנה לכל הליטרלים האלה ערך $TRUE$ (אחרת לפחות אחד מהם הוא שלילה של אחר - וזה לא אפשרי, כי אין ביניהם קשתות). השמה זו מספקת את φ. ■

מסקנה 3.16

$$CNF-SAT \leq_p IS$$

ודי קל להראות את הכיוון השני, לכן אם $CNF-SAT \notin P$ אז גם $IS \notin P$, וההפך.

הגדרה 3.17 שפה L נקראת NP-קשה אם לכל $L' \in NP$ מתקיים $L' \leq_p L$.
אם $NP \ni L$, אז אומרים ש-L היא NP-שלמה.

בשיעור הבא נוכיח:

משפט 3.18 $CNF-SAT$ היא NP-קשה (וגם שלמה).

30/5/2012

מטרתנו היום - להראות שכל שפה $L \in NP$ קיימת רדוקציה ל-$CNF-SAT$. ממה שהוכחנו בשיעורים הקודמים ובתרגולים, נוכל להסיק כי $CNF-SAT \in NPC$. על כן חשיבותה הגדולה של בעיה זו - אם היא ב-P, $P=NP$. אם היא לא ב-P, אז גם את שאר הבעיות שראינו - $IS, CLIQUE, VC, SUBSET-SUM$ - לא ניתן לפתור בזמן פולינומי.

הגדרה 3.19 תהא M מ"ט לא דטרמיניסטית מעל א"ב קלט Σ וא"ב עבודה Γ.
נגדיר את השפה W_M להיות:

$$W_M = \{ \langle x \rangle \# \langle y \rangle | \, x \text{ is an input for } M, \, y \text{is an accepting run of } M \text{on } x \}$$

אבחנה: עבור $x \in \Sigma^*$, $\exists y \;\; \langle x \rangle \# \langle y \rangle \in W_n \Leftrightarrow x \in L(M)$.

טענה 3.20 $W_M \in P$.

הוכחה: קל לראות שקיימת מ"ט V_M אשר בהנתן $\langle x \rangle \# \langle y \rangle$ מוודאת כי $\langle y \rangle$ הוא קידוד של ריצה חוקית מקבלת של M על הקלט x, כך ש-V_M רצה בזמן פולינומי ב-$|\langle x \rangle \# \langle y \rangle|$[27]. ■

הגדרה 3.21 תהא $B-A_{TM}$ שפת שלשות:

$$B-A_{TM} = \left\{ \left(\langle M \rangle, \langle w \rangle, 1^t \right) \middle| \, \exists y, \, |y| \leq t \text{s.t. } M \text{accepts } w\#y \text{in at most } t \text{steps} \right\}$$

כאשר M היא מ"ט דטרמיניסטית, w היא קלט עבור M, 1^t היא שרשור של t "1"-אים.

טענה 3.22 $B-A_{TM} \in NP$.

הוכחה: נבנה מ"ט לא דטרמיניסטית המזהה את $B-A_{TM}$:
בהנתן שלשה $(\langle M \rangle, \langle w \rangle, 1^t)$, המכונה תכתוב באופן לא דטרמיניסטי קידוד $\langle y \rangle \# \langle y \rangle$ ותסמלץ ריצה של M על הקלט $w\#y$ במשך t צעדים לכל היותר - אם M המסומלצת קיבלה, המכונה שלנו תקבל, אחרת תדחה (אם M לא תעצור, המכונה שלנו תעצור אותה אחרי t הצעדים ותדחה).
כמה זמן כל התהליך הזה יקח?
כתיבת הקידוד של y - פולינומי ב-t.
סימלוץ כל צעד בריצה של M על הקלט - פולינומי באורך הקונפיגורציה.
אורך הקונפיגורציה ההתחלתית - פולינומי באורך הקלט (השלשה).
אזי הכל פולינומי באורך הקלט, ולכן המכונה רצה בזמן פולינומי, כלומר $B-A_{TM} \in NP$. ■

האם אפשר להגיד על $B-A_{TM}$ משהו חזק יותר?

טענה 3.23 $NPC \ni B-A_{TM}$.

[27] תהליך הוידוא הזה כולל בדיקה בכל קונפיגורציה כי היא קונפיגורציה עוקבת חוקית של הקודמת, ולסיום יש לבדוק כי הקונפיגורציה הסופית היא מקבלת. את כל הפעולות הללו ניתן לבצע בזמן פולינומי בקלט.

הוכחה: נראה כי $B-A_{TM}$ היא NP-קשה, על ידי כך שניקח איזשהיא שפה $L\in NP$, ונראה כי קיימת רדוקציה פולינומית מ־L ל־$B-A_{TM}$:

אזי, קיימת מכונת טיורינג לא דטרמיניסטית הרצה בזמן פולינומי $c\cdot n^k\geq$ על קלטים באורך n ומזהה את L - נסמנה ב־M.

הרדוקציה מ־L ל־$B-A_{TM}$ תפעל כדלהלן:

בהנתן קלט x, הרדוקציה תחזיר את השלשה $(\langle V_M\rangle,\langle x\rangle,1^t)$, כאשר $t=\left(c\cdot|x|^k\cdot\left(|x|+c\cdot|x|^k\right)\right)^{10}$.

הערה 3.24 לכל L, תהיה מכונה V_M שונה שתזהה אותה, ולכן הרדוקציה תהיה שונה.

1. הרדוקציה עוצרת בזמן פולינומי: שכן V_M היא מכונה קבועה, לכן הכתיבה שלה היא בזמן פולינומי, וברור שהשאר גם כן.
2. אם $x\in L$, אז ל־M יש ריצה מקבלת על x בזמן $c\cdot|x|^k\geq$. תהא y הקידוד של תיאור ריצה זו - כלומר y מתאר ריצה שמספר הקונפיגורציות שלה הוא $c\cdot|x|^k\geq$. נשאל, מהו האורך של קונפיגורציה? לא ברור, אבל האורך הכי גדול של קונפיגורציה שנוכל להגיע אליו הוא $|x|+c\cdot|x|^k$. אם נכפול מספר זה במספר הקונפיגורציות המקסימלי נגיע לחסם על האורך של y - $c\cdot|x|^k\cdot\left(|x|+c\cdot|x|^k\right)$. אזי, V_M תקבל את הקלט $x\#y$ תוך זמן $t\geq\left(c\cdot|x|^k\cdot\left(|x|+c\cdot|x|^k\right)\right)^{10}\geq$[28].
3. אם $(\langle V_M\rangle,\langle x\rangle,1^t)\in B-A_{TM}$, אז $x\in L$: אם התנאי מתקיים, אזי קיים y כך ש־V_M מקבל את $x\#y$, ולכן $x\in L(M)=L$.

■

הערה 3.25 ההוכחה מעלה היא מסוג די שונה ממה שנראה מעכשיו והלאה, ומהרדוקציות הפולינומיות שראינו עד עכשיו. שאר הדברים הם טכניים.

משפט $Cook-Levin$**:**

משפט 3.26 $CNF-SAT\in NPC$.

הוכחה: נראה רדוקציה מ־$B-A_{TM}$ ל־$CNF-SAT$.

טענה 3.27 יהי P פרדיקט[29] על משתנים בוליאניים $x_1,...,x_k$. אזי יש נוסחת $CNF-SAT$ φ על המשתנים הנ"ל המייצגת את P, ומספר המופעים של ליטרלים בה הוא לכל היותר $k\cdot2^k\geq$.

הוכחה: עבור השמה α למשתנים, נגדיר:

$$C_\alpha=y_1\vee y_2\vee...\vee y_k$$

כאשר:

$$\forall i:\ y_i=\begin{cases}\overline{x}_i & \alpha(x_i)=1\\ x_i & \alpha(x_i)=0\end{cases}$$

אז לכל השמה β מתקיים:

$$C_\alpha(\beta)=\begin{cases}0 & \beta=\alpha\\ 1 & \beta\neq\alpha\end{cases}$$

נגדיר:

$$\varphi:=\bigwedge_{\alpha:\ P(\alpha)=0}C_\alpha$$

[28] 10 שרירותי גדול מספיק, שכן נקבל פולינום באורך הקלט.

[29] פרדיקט - פונקציה של משתנים בוליאניים, שכאשר מזינים לתוכה ערכים למשתנים, מקבלים הערכה.

אז:

$$\varphi(\beta) = \begin{cases} 1 & P(\beta) = 1 \\ 0 & P(\beta) = 0 \end{cases}$$

וכן φ אכן מכילה $k \cdot 2^k \geq$ מופעים של ליטרלים, כנדרש. ■

מסקנה 3.28 כל עוד יש פרדיקט על מספר קטן של משתנים, אז אפשר לבטא אותו בעזרת נוסחת $CNF - SAT$.

4/6/2012

לכן נגדיר רדוקציה מ־$B - A_{TM}$ ל־$CNT - SAT$, אשר בהנתן שלשה $(\langle M \rangle, \langle w \rangle, 1^t)$, מחזיר הסגרים מכל קב' משתנים בוליאנים אשר ניתן לסיפוק אמ"מ M עוצרת ומקבלת תוך $t \geq$ צעדים על הקלט, כלומר אמ"מ קיים y, $|y| \leq t$ כך ש־M מקבלת את $w\#y$ תוך t צעדים או פחות.

בעצם נראה רדוקציה המייצרת אוסף פרדיקטים, כי אז בעזרת הטענה נוכל לעבד את הפלט להסגרים שקולים.

כיצד נעשה זאת? נרצה לתרגם את הקונפיגורציות של M בריצתה על $w\#y$ במשך t צעדים לפרדיקטים.

נחשוב על ריצה של t[30] קונפיגורציות, כאשר אם המכונה עוצרת לפני, נחליט שהקונפיגורציה הבאה תהיה שווה לקונפיגורציה הקודמת. על כן, בלי הגבלת הכלליות ניתן לדבר על t קונפיגורציות. הקונפיגורציה עצמה אינה אינסופית, שכן אנו מסתכלים רק על t צעדים, על כאן הראש הקורא\כותב יוכל להגיע רק t תאים ימינה ושמאלה על הסרט, על כן נוכל לייצג כל קונפיגורציה באופן סופי.

הרדוקציה תצור שלושה סוגי משתנים:

- משתני הסרט: לכל זמן $s \in \{0, 1, ..., t\}$, לכל אות $\alpha \in \Gamma$, ולכן מקום על הסרט $i \in \{-t, ..., t\}$, נתאים משתנה בוליאני $x_{S,\alpha,i}$, אשר הוא $\mathbb{T}$ אמ"מ "בזמן s כתובה האות α במקום ה־i".

נותר לייצג את מה שהראש עושה:

- משתני מיקום הראש: לכל זמן $s \in \{0, 1, ..., t\}$ ולכל $i \in \{-t, ..., t\}$ נתאים משתנה בוליאני $h_{s,i}$ אשר הוא $\mathbb{T}$ אמ"מ "בזמן s הראש נמצא במקום ה־i".
- משתני (מצב) הראש: לכל $s \in \{0, 1, ..., t\}$ ולכל $q \in Q$ (של M), נתאים משתנה בוליאני $g_{s,q}$ אשר הוא $\mathbb{T}$ אמ"מ "מצב הראש בזמן s הוא q".

בעזרת השמה למשתנים הבוליאניים האלו, אפשר לייצג כל ריצה מקבלת של המכונה על קלט מהצורה $w\#y$ (למעשה אפשר לייצג ריצה על כל קלט) שלוקחת לכל היותר t צעדים.

עם זאת, יש הרבה השמות אפשריות למשתנים הבוליאניים האלה שלא מייצגות ריצה חוקית. על כן, נוסיף פרדיקטים שיסופקו אמ"מ זוהי אכן ריצה חוקית של המכונה M. לא נכתוב את כולם אבל נתאר אותם, מכיוון וזהו תהליך טכני בלבד:

- סוג ראשון ־ "מניעת כפילויות":
  - סרט: "לכל זמן $s \in \{0, 1, ..., t\}$ לכל מיקום $i \in \{-t, ..., t\}$ על הסרט, ולכל שתי אותיות שונות $\alpha \neq \beta \in \Gamma$:

$$\overline{x}_{s,\alpha,i} \vee \overline{x}_{s,\beta,i}$$

  - מיקום הראש:

$$\forall s \in \{0, 1, ..., t\},\ \forall i \neq j \in \{-t, ..., t\} : \overline{h}_{s,i} \vee \overline{h}_{s,j}$$

  - מצב הראש: כנ"ל.
- סוג שני ־ "קיום מצב":
  - סרט: "בכל מקום בסרט ובכל זמן תהיה כתובה איזושהיא אות":

$$\forall s \in \{0, 1, ..., t\},\ \forall i \in \{-t, ..., t\} : \bigvee_{\alpha \in \Gamma} x_{s,\alpha,i}$$

[30] למען האמת זה $t + 1$ אבל לא נהיה קטנוניים.

- מיקום ומצב הראש[31].

- סוג שלישי - "חוקיות המעברים":

- עבור כל שיוויון מהצורה $\delta(q_1,\alpha)=\delta(q_1,\beta,R)$, יהיה פרדיקט:
"אם $x_{s,\alpha,i}$ וגם $h_{s,i}$ וגם $g_{s,q}$, אזי $x_{s+1,\beta,i}$ וגם $h_{s+1,i+1}$, וגם g_{s+1,q_2} (פרדיקט על 6 משתנים בוליאניים, לכן מטענה העזר אין בעיה להעביר אותו לצורת CNF, והאורך של מה שנקבל הוא קצר מספיק).

- נשאר ולא נפרט:

- "אחרי עצירה נשארים באותו מצב".
- $g_{t,q_{acc}}$ - אם מסתפק אכן מדובר בריצה מקבלת.
- תנאי ההתחלה:
  - $h_{0,0}\wedge g_{0,q_0}$.
  - "בתחילת הריצה, הקלט (ורק הוא) רשום על הסרט" - כלומר על הסרט מופיע $w\#y$.
- אות אינה משתנה אם הראש לא נמצא מתחתיה: $x_{s,\alpha,i}\wedge\overline{h}_{s,i}\Rightarrow x_{s+1,\alpha,i}$

על כן, זוהי אכן רדוקציה, והיא אכן עובדת בזמן פולינומי, שכן מספר המשתנים הוא פולינומי ב־t (ואולי גם באורך קידוד המכונה), ולכן פולינומי באורך הקלט, שכן t מיוצג בצורה אונרית. לגבי מספר הפרדיקטים - נשאר לקורא הנלהב. ■

3.4 היררכיית הזמן

אנחנו לא מכירים בעיה טבעית חישובית שיודעים שהיא ב־P, אך לא יודעים שאי אפשר לפתור אותה בזמן ריבועי.
בשיעור הזה ניצור בעיות "לא כל כך טבעיות", כדי להוכח שציור ההיררכיות אכן בעל חשיבות. המסקנה שנגיע אליה - "כשנותנים יותר זמן, מקבלים יותר כוח חישובי".
איך בכלל מראים שלא ניתן להכריע בעיה בפחות מ־t^n? ראינו "משהו כזה" כשהראנו שבעיות אינן כריעות - הנחנו בשלילה שכן, לקחנו את המכונה ה"מדומיינת" שמכריעה, ואז בהנתן שאלה למכונה הזו, נסמלץ בעזרת מכונה אוניברסלית כדי לקבל את התשובה, אבל נגרום לה להחזיר תשובה הפוכה כדי לגרום לה לטעות. את הרעיון הכללי ניישם גם כאן.
מכיוון ואנו מדברים ברזולוציה יותר גבוהה - כלומר, הכרעה תוך זמן מסויים, יש לבדוק כמה זמן לוקח לעשות סימולציה לריצה של מכונה טיורינג.

הגדרה 3.29 $t:\mathbb{N}\to\mathbb{N}$ נקראת חשיבה בזמן, אם $t(n)\geq\Omega(n\log n)$, וגם קיימת מ"ט אשר בהנתן n, מחשבת את $t(n)$ בזמן $O(t(n))$.

דוגמאות:

- $t(n)=n\log n$[32].
- $t(n)=n^3$.
- $t(n)=2^n$.

כל סיבוכיות "נורמלית" היא פונקציה חשיבה.

טענה 3.30 קיימת מ"ט S אשר בהנתן $(\langle M\rangle,\langle w\rangle,t)$ מחשבת את הקונפיגורציה של M בריצתה על w במשך t צעדים ומקיימת:
S רצה בזמן $\geq\left(t\cdot\left(|\langle M\rangle|^3+|\langle M\rangle|\cdot\log t\right)\right)$ O על קלטים שעבורם $t\geq|\langle M\rangle|+|w|$[33].

לפני שנוכיח, נזכר במשהו משיעורי הבית:

$$L=\left\{w\in\{a,b\}^*\middle|\,\#_a(w)=\#_b(w)\right\}$$

הוכחנו כי $L\subseteq TIME(n\log n)$.

[31] אפשר לדאוג להם באותם פרדיקטים.
[32] $\log n$ מחשבים ע"י ספירת הספרות בייצוג הבינארי של n.
[33] ואז לא צריך להכניס את $|w|$ לזמן הריצה של S.

יש שני פתרונות לבעיה. הפתרון המועדף על המרצה[34]:
מחזיקים מונה, הראש סורק את הסרט משמאל לימין, מזיזים את המונה בכל צעד כך שיהיה משמאל לאות שקוראים עכשיו (דורסים את מה שכבר נקרא), ומעלים ב־1 אם קראנו a, ומורידים ב־1 אם קראנו b. הבעיה היחידה - היא דורסת את המילה. אפשר לנסות לא לדרוך ע"י דחיפה של האות שמאלה דרך הקאונטר - פעפוע. כך ניתן לשמור על המילה המקורית, אבל אם צריך להוסיף עוד ספרה למונה צריך להזיז תא הרישא או הסיפא וזה יכול לקחת זמן...
דרך פשוטה לפתור את הבעיה הזו:
נחזיק את המונה יחד עם האותיות של המילה - מוסיפים לא"ב העבודה את האותיות $\frac{a}{1}, \frac{a}{1}, \frac{b}{0}, \frac{b}{1}$, ואז התאים שכבר עברנו עליהם יכילו גם את האות וגם את הערך של המונה, ואז אין בעיה להוסיף תאים למונה. אז אין צורך להזיז תאים ימינה ושמאלה. גם אז צריך במעבר לקריאת האות החדשה להעביר את המונה תא אחד שמאלה, אבל עושים את זה בקלות. אז חוזרים לתחילת המונה, קוראים אות חדשה, וחוזר חלילה. אורך המונה הוא לא יותר מ־$\log n$, אז לא צריך יותר מזה כדי להעלות אותו ו־$\log n$ כדי להזיז אותו.
זו שיטה כללית שאפשר לעשות איתה הרבה דברים - אם רוצים למשל להזיז את הראש שמאלה ולהזיז את המונה, אפשר לעשות זאת, ועדיין הצעד הוא לוג מספר הספרות במונה.
נשתמש ברעיון הזה לסימולציה להוכחת הטענה:
איור
ניתן לחשב את $\delta(q,a)$ בהנתן $(\langle M\rangle, q, a)$ בזמן $\geq O\left(|\langle M\rangle|^3\right)$[35]. **הוכחה:** ניתן לממש את S כך:
בהנתן $(\langle M\rangle, \langle w\rangle, t)$, S תשמור מונה אותו היא תוריד אחרי כל שלב בסימולציה, ויתחיל מהמספר t. S תעצור כאשר $t=0$.
בכל צעד, S תשתמש באלגוריתם שתיארנו כדי לחשב את הקונפיגורציה הבאה, ותמשוך את $\langle M\rangle$ ו־q "מתחת" למחרוזת שעל הסרט:
המחרוזת $\langle M\rangle, q\in Q, \langle a\rangle$ תופיע על הסרט, ומימין ומשמאל המילה עליה היא רצה, משמאל צמוד ל־$\langle M\rangle$ יופיע המונה. נרצה שהמצב של M והקידוד ישאר ליד המילה הבאה - נשנה את הא"ב כך ש־$\langle M\rangle, q$ יופיעו מתחת לקלט:
איור
הראש של M תמיד נמצא על הקידוד של האות הבאה לקרוא, הראש של S "נודד" כפי שתיארנו מעלה.

הערה 3.31 נשים לב כי אם המחרוזת שאנחנו רוצים "לסחוב" איתנו היא קבועה, אין בעיה להעביר אותה ללא שימוש בא"ב כפול\תאים כפולים - שומרים מרווח קבוע למחרוזת בין האותיות. נשים לב כי בבעיה זו אין לנו באמת צורך להשתמש בשיטת הא"ב הכפול - המקום שצריך המונה בטוח לא גדל, וכן אורך $\langle M\rangle$ ו־q קבועים. על כן מספיק להזיז השרשרת בכל צעד.

נשאר לבדוק אם הזמן שטענו שלוקח לסימולציה לעבוד הוא נכון או לא:
הצעד לוקח $|\langle M\rangle|^3$ כפול t צעדים.
שינוי המונה - $\log t$, צריך לעשות t פעמים - על כן $t\cdot\log t$.
העברה של המחרוזת - $(|\langle M\rangle|+\log t)\cdot|M|$ - אורך המחרוזות כפול מספר התאים שיש להזיז את המחרוזת בכל צעד - שהוא אורך הקידוד של w על הסרט של S. גודל זה הוא לכל היותר $|\langle M\rangle|$. ■

משפט היררכיית הזמן:

משפט 3.32 תהי $t:\mathbb{N}\to\mathbb{N}$ חשיבה בזמן.
אזי, יש $L\in TIME[t(n)]$ שאינה חשיבה בזמן $o\left(\frac{t(n)}{\log t(n)}\right)$.

למשל, יש שפה שאפשר להכריע בזמן n^2, אבל אי אפשר להכריע בזמן:

$$\frac{n^2}{\log^2 n}=o\left(\frac{n^2}{\log(n^2)}\right)$$

הוכחה: רעיון: נבקש ממכונה לסמלץ, ואז נהפוך את התשובה.
נגדיר את השפה הבאה:

$$L:=\left\{(\langle M\rangle,\langle w\rangle)\,\middle|\,M\text{doesn't accept }(\langle M,w\rangle)\text{in}\leq\frac{t(n)}{|\langle M\rangle|^3+|M|\cdot\log t(n)}\text{steps, where }n=|\langle M,w\rangle|\right\}$$

תחילה, נראה $L\in TIME[t(n)]$: בהנתן $\langle M,w\rangle$, נגדיר מ"ט T כדלהלן:
בהנתן $\langle M,w\rangle$, T תחשב את $t=\left\lfloor\frac{t(n)}{|\langle M\rangle|^3+|\langle M\rangle|\cdot\log t(n)}\right\rfloor$. עתה, T תסמלץ את הריצה של M על $\langle M,w\rangle$ בעזרת S מהטענה הקודמת במשך t צעדים. אם M לא קיבלה במהלך הסימולציה, T תחזיר "כן", אחרת T תדחה.

[34] פתרון נוסף מופיע בספר של ספיצר.
[35] אפשר להוריד את החסם הזה.

אזי $L(T) = L$.

זמן הריצה[36]: $t(n) \geq t \cdot \left(|\langle M \rangle|^3 + |\langle M \rangle| \cdot \log t\right)$[37].

חלק שני (שמו עדיין לא ידוע):

תהא $R : \mathbb{N} \to \mathbb{N}$, ונניח בשלילה כי $R(n) = o\left(\frac{t(n)}{\log t(n)}\right)$, ויש מ"ט M_0 שמכריעה את L בזמן $R(n)$. עבור n גדול מספיק, מתקיים:

$$R(n) < \frac{t(n)}{|\langle M_0 \rangle|^3 + |\langle M_0 \rangle| \cdot \log t(n)}$$

על קלט w באורך גדול מ־n, יתקיים שאם M_0 דוחה את הקלט $\langle M_0, w \rangle$, אזי $\langle M_0, w \rangle \in L$, וזו סתירה, שכן M_0 אמורה לקבל את L, לכן אם $\langle M_0, w \rangle \in L$, M_0 אמורה לקבל את הקלט הזה. אם M_0 מקבלת את $\langle M_0, w \rangle$, אזי היא עושה זאת ב־$R(|\langle M_0, w \rangle|)$ צעדים, ולכן $\langle M_0, w \rangle \notin L$, בסתירה. ■

בתרגול נראה את משפט ההיררכיה במרחב, שמאוד דומה להוכחה של משפט ההיררכיה בזמן.

and now, for something completly different!

[36] זהו זמן הריצה של הסימולציה, צריך להוסיף גם את הזמן של חישוב t, אבל "לא נכנס לזה".

[37] הכל מצטמצם, על כן ההגדרה ה"מוזרה" של השפה L.

4 מודלים נוספים של מכונת טיורינג

עד כה דנו (בעיקר) במכונת טיורינג עם סרט אחד, המקבלת את הקלט על הסרט. לא התחלנו אפילו לגרד את האפשרויות לגבי המכונה הזו, ולגבי הגרסא האי דטרמיניסטית שלה.

ועם זאת - נמשיך הלאה.

את המודל הזה אפשר לשנות כיד הדמיון הטובה, ולקבל כל מיני מודלים מעניינים. לעיתים מודלים אלו מעניינים כשלעצמם, ולעיתים מגיעים אליהם תוך כדי נסיון לפתור בעיות במודל הבסיסי.

למשל, מודל תקשורת - מודל בעל שתי מכונות שיש קשר ביניהן, המנסים לפתור בעיה הקשורה לקלט של שתיהן. ברגע שמדברים על מודל של תקשורת, אפשר לדבר על רעש - מה אם לא כל ביט שנשלח מצד אחד מגיע "כמו שצריך" לצד השני? זהו סוג אחד של שאלה חישובית מאוד בסיסית וחשובה ליום יום, ועליה לא דיברנו בכלל.

מודל נוסף שגם קשור לתקשורת - אם יש שני צדדים וקו תקשורת ביניהם, אבל הם לא מחשבים פונקציה משותפת, אלא אחד טוען טענה מסויימת מתמטית נניח, או שהוא בעל כרטיס אשראי מסויים, והצד השני צריך לוודא שהטענה הזו נכונה. אפשר לשאול כמה תקשורת צריך כדי לוודא שהטענה נכונה? מה קורה כשיש רעש? כל הדברים האלה יוצרים סט ענק של שאלות.

אפשר לדבר על מודלים רבים נוספים. בתרגול נראה מודל שאין ממש קו תקשורת אלא אורקל - מכונה בעלת "סרט קסם" נוסף - המכונה כותבת עליו שאלה עם סימן שאלה, וכאשר נכתב סימן השאלה, מופיע עליה התשובה. אפשר לשאול, מה אפשר לחשב על מכונה ורצה בזמן פולינומי, בעלת אורקל שיכול לפתור למשל $3SAT$? מה קורה עם מכונה לא דטרמיניסטית עם אורקל?

היום אנחנו נדבר על שינוי אחד ספציפי במודל:

4.1 הוספת עצה - $Advice$

זהו סרט נוסף וראש נוסף, שהוא ראש שיכול רק לקרוא (אינו יכול לכתוב), ויכול ללכת ימינה בלבד[38]. על הסרט השני כתובה עצה:

$$r = r_1, r_2, r_3, ...$$

נחשוב על מצב בו העצה היא אינסופית, כדי שלא נתעסק עם השאלה "כמה ביטים יש לעצה". ובאופן פורמלי:

הגדרה 4.1 מ"ט עם עצה היא מ"ט עם שני סרטים, כאשר אחד מהם הוא סרט קריאה חד פעמי, אשר בתחילת הריצה רשומה עליו מחרוזת בינארית אינסופית.

אזי, שוב צריך לשאול את השאלות לגבי זמן ריצה:

הגדרה 4.2

$$t_M(n) = \max_{|x| \leq n,\ r} \{\text{Runtime of } M \text{on } x \text{with the advice } r\}$$

הערה 4.3 כביכול יש כאן "אינסוף" עצות, אבל כאלו הנבדלות בביט שלא קוראים אותו בעצם לא משנות דבר.

אז, מה אפשר לעשות עם מכונה שכזו? אפשר להגדיר סוגים מעניינים שונים של מחלקות.

הגדרה 4.4 תהא M מכונת טיורינג עם עצה. M היא אלגוריתם ווידוא לשפה L, אם מתקיים:

$$\forall x\ x \in L \Leftrightarrow \exists r\ s.t.\ M \text{accepts } x \text{with advice } r$$

אפשר לחשוב על r כעזר למכונה לקבוע האם קלט מסויים בשפה. זה מזכיר לנו כאן דבר שראינו בעבר - "עד" במכונות אי דטרמיניסטיות.

נניח שלשפה L כלשהיא יש אלגוריתם וידוא הרץ בזמן פולינומי $\left(\geq O\left(|x|^k\right)\right)$. מה ניתן אז להגיד על L? אזי, $L \in NP$, שכן ניתן לבנות מכונה אי דטרמיניסטית שריצותיה הן ריצות חוקיות על M עבור איזושהיא עצה - כלומר, בהנתן אלגוריתם וידוא, המכונה הלא דטרמיניסטית, בהנתן x, תרשום עצה (מספר פולינומי של ביטים - $|x|^k$) בצורה לא דטרמיניסטית, ואז תסמלץ את ריצת M על הקלט x עם העצה שנכתבה. אזי, אם $x \in L$, קיימת ריצה בה המכונה האי דטרמיניסטית תקבל את הקלט, שכן קיימת עצה שבריצה עליה ועל x, M תקבל.

אזי, מודל זה "תופס" בצורה קלה את המכונה האי־דטרמיניסטית. יש עוד הרבה דברים אחרים שאפשר לעשות איתה:

[38] זה רלוונטי רק כאשר מקום מוגבל - אחרת אפשר פשוט להעתיק את הכתוב ולקרוא אותו כש"בא לנו".

4.1.1 אלגוריתמים רנדומיים

הגדרה 4.5 מ"ט רנדומית היא מ"ט עם עצה, כאשר העצה נקבעת בתחילת הריצה להיות הייצוג הבינארי של מספר α הנבחר בהתפלגות אוניפורמית מתוך הקטע $[0,1)$.

הערה 4.6 הוספת כל דבר להגדרה הזו מסתירה מורכבות שלא נדון בה, על כן, נסתפק בהגדרה זו.

אזי, כל ריצה של מכונה שכזו היא מאורע במרחב הסתברות מסויים. אפשר לדבר למשל על המחלקה - $RTIME\left[f\left(n\right)\right]$ השפות שניתן לזהות ע"י אלגוריתם רנדומי בזמן $f\left(n\right)$. אבל בשביל הדיון הזה, צריך להגדיר באופן פורמלי:

הגדרה 4.7 $RTIME\left[f\left(n\right)\right]$ היא אוסף השפות L עבורן קיים אלגוריתם ווידוא M שרץ בזמן $O\left(f\left(n\right)\right)$, כך ש-

$$\forall x \in L \; Pr_r\left(M\text{accepts } x\text{with advice } r\right) \geq \frac{1}{2}$$

ובאופן דומה:

הגדרה 4.8

$$RP := \bigcup_{k\geq 1} RTIME\left[n^k\right]$$

הערה 4.9 למען האמת, הגדרה הזו דומה יותר להגדרה של NP ולא של P. ולכן אפשר להגדיר:

הגדרה 4.10

$$co-RP = \left\{L \middle| \overline{L} \in RP\right\}$$

האלגוריתם הנאיבי ל-SAT לא מראה שהוא ב-RP - שכן אם יש רק השמה מספקת אחת, אזי ההסתברות שבהנתן r, M תקבל, הינה $\frac{1}{2^n}$.
מה כן?

$$COMPOSITE \in RP$$

בעזרת האלגוריתם של MR - משתמש בביטים הרנדומיים של העצה כדי להגדיר את A, בודק על זה כל מיני בדיקות, ואם המספר ראשוני תמיד נדע, אחרת יהיה "עד" לכך שהמספר הוא $composite$, והסיכוי של MR להגיד שהוא אכן $composite$ הוא $\frac{1}{2}$.
בשנת 2002 הוכיחו כי $COMPOSITE$ ב-P.
אנחנו לא יודעים אם $RP = P$, ברור כי $P \subseteq RP$, שכן אפשר לחשוב על מכונה שפשוט מתעלמת מהעצה. להבדיל מהשאלה על NP, משערים היום שהמחלקות הללו שוות.
נביט במחלקה:

$$ZPP := RP \cap co-RP$$

אלגוריתם RP חזק יותר מאלגוריתם וידוא (שכן רב המחרוזות מההגדרה בשפה), ועל כן $RP \subseteq NP$. האמונה הרווחת היא ש-$RP = P$.
כמו כן, אנו יודעים כי $co-NP \supseteq co-RP \supseteq P$.
ישנה גם מחלקה בשם BPP של שפות שיש אלגוריתם שיכול לזהותם, ולא טועה ביותר מ-$\frac{1}{3}$, אבל עליה לא נדבר.
הנושא הראשון של היום - נראה בעיה שהיא ב-RP, ואנו לא יודעים איך לפתור אותה בזמן פולינומי. לא יהיו לנו זמן להראות זאת, אבל זו בעיה שהיא יחסית טבעית, והיא חשובה ושימושית:

4.2 בעיית בדיקת זהות עבור נוסחאות - Identity Testing for Arithmetic Forumlas

חשובה, כי היא עולה כאשר מנסים להבין הכלה של מחלקות, ושימושית, כי היא הרבה פעמים שימושית בחיים האמיתיים.
אנו אמורים להתחיל מהגדרת הבעיה, אבל, נתחיל מלדבר על פולינומים.

4.2.1 פולינומים

בעיקר כאשר מדברים על רנדומיות, אבל בהרבה מאוד קונטסקטים אחרים, כאשר מתחילים לשאול שאלות רציניות במדעי המחשב, עולים הרבה שאלות אלגבריות - והרבה פולינומים.

הגדרה 4.11 מונום במשתנה x, הוא ביטוי מהצורה x^i עבור $i \in \mathbb{N} \cup \{0\}$.

הגדרה 4.12 פולינום הוא קומבינציה לינארית של מונומים.

למשל, $p(x) = 12 \cdot x^7 - 5.3 \cdot x^2 + 3$. אנו נתעניין בעיקר במקרים בהם המקדמים בפולינום הם שלמים.

הגדרה 4.13 דרגה של פולינום $p(x)$ היא הדרגה הגבוהה ביותר של המשתנה x המופיעה בפולינום.

למשל, $deg(p) = 7$.
אפשר לעשות הרבה מאוד דברים עם פולינומים - קונובלוציה, אינטגרל, גזירה...
הדבר אולי הכי בסיסי והכי שימושי שאפשר לעשות עם פולינומים הוא הערכה:

הגדרה 4.14 הערכה של פולינום $p(x)$ שניתן לסמנה ב-$p(x)|_{x=i} = p(i)$ היא התוצאה המתקבלת בעת הצבת i בפולינום במקום x.

הגדרה 4.15 פולינום האפס הוא $p(x) = 0$. דרגתו מוגדרת להיות $-\infty$.
דבר המבדיל אותו משאר הפולינומים הוא שבכל נקודה, ההערכה שלו שווה לאפס.

הגדרה 4.16 שורש של פולינום $p(x)$ הוא מספר שההערכה של הפולינום בו היא 0.

ראינו בקורסים מתמטיים קודמים כי:

טענה 4.17 יהי $p(x)$ פולינום מדרגה $k \geq$, שאינו פולינום האפס. אזי, מספר השורשים של p הוא $\leq k$.

אם נרצה לבדוק האם פולינום נתון הוא פולינום האפס, בהנתן שיש לנו מכונה שיודעת להעריך את הפולינום בנקודה נתונה, כל מה שצריך לעשות הוא להזין $k+1$ ערכים, ולבדוק כמה הערכות מקבלות אפס - אם $k+1$ ההערכות הן אפס, אזי זהו פולינום האפס, אחרת לא.
ומה קורה אם נותנים לנו גישה אחת בלבד? אין לנו יכולת לדעת בביטחון.
ומה אם יש לנו מנגנון רנדומי?

בדיקת זהות לאפס (רנדומית): נניח $deg(p) \leq k$. נבחר מספר רנדומי α מתוך הקבוצה $\{1, 2, 3, ..., 2k\}$, ונעריך את $p(\alpha)$. אם $p(\alpha) \neq 0$ נחזיר "$p \neq 0$", אחרת נחזיר "$p = 0$".
תמיד כאשר הפולינום הוא פולינום האפס, נחזיר כי הוא פולינום האפס תמיד. מהו הסיכוי שלנו לטעות, כלומר, מה הסיכוי שנחזיר כי הוא פולינום האפס כאשר הוא אינו פולינום האפס?

$$p \equiv 0 \Rightarrow \text{the answer is always "}p = 0\text{"}$$
$$p \neq 0 \Rightarrow Pr_{\alpha}[\text{answer "}p \neq 0\text{"}] \geq \frac{1}{2}$$

ע"י חזרה על האלגוריתם, אפשר להקטין את הטעות כרצוננו (ע"י התאמת מספר הפעמים לחזרה).
אבל, יש לנו רק גישה אחת. מה נעשה? נבחר את α להיות מספר מתוך תחום הרבה יותר גדול, ואז גם כן ניתן להקטין את ההסתברות לטעות, שכן אם הפולינום אינו פולינום האפס, יש רק k ערכים בהם הוא אמור לקבל 0[39].

הגדרה 4.18 מונום ב-n משתנים, $x_1, x_2, ..., x_n$ הוא ביטוי מהצורה:

$$x_1^{\beta_1} \cdot x_2^{\beta_2} \cdot ... \cdot x_n^{\beta_n},\ \beta_1, ..., \beta_n \in \mathbb{N} \cup \{0\}$$

הגדרה 4.19 באופן דומה, ניתן להגדיר פולינום במספר משתנים להיות צירוף של מונומים.

[39] לכאורה, היינו יכולים להגריל מספר רנדומי מתוך קטע רציף, ואז הסיכוי לטעות היה אפס. אולם, אנו חותרים לאלגוריתם שניתן להריץ במחשב, ולא ניתן להגריל מספר שכזה בעזרת מחשב, שכן הייצוג של מספר שכזה הינו אינסופי.

למשל:

$$p(x_1, x_2, x_3) = x_1 \cdot x_2 + 2 \cdot x_3^{17} - 3 \cdot x_1 \cdot x_2^4 \cdot x_3^{13}$$

הגדרה 4.20 הדרגה של מונום במספר משתנים מוגדרת להיות:

$$deg\left(x_1^{\beta_1} \cdot ... \cdot x_n^{\beta_n}\right) = \sum_{i=1}^{n} \beta_i$$

הגדרה 4.21 הדרגה של פולינום במספר משתנים מוגדרת להיות:

$$degp(x) = maxdeg(m)$$

כאשר m הוא מונום המרכיב את p עם מקדם שאינו 0.

4.2.2 נוסחאות

הגדרה 4.22 נוסחא אריתמטית מעל המשתנים $x_1, ..., x_n$ היא אחת מהאפשרויות הבאות:

1. -1 ,1 ,0.
2. x_i, כאשר $1 \leq i \leq n$.
3. $(\varphi_1) + (\varphi_2)$, $(\varphi_1) \cdot (\varphi_2)$, כאשר φ_1, φ_2 הן נוסחאות.

מה שטוב בנוסחאות אריתמטיות - ניתן להציג בעזרתם פולינומים!
נביט למשל בנוסחא האריתמטית הבאה:

$$\varphi = (x_1 + x_2) \cdot (x_2 + x_3)$$

נוסחא זו מייצגת פולינום בשלושה משתנים, שהוא:

$$p(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2^2 + x_1 \cdot x_3 + x_2 \cdot x_3$$

ואפילו עוד דוגמא:

$$\varphi = (x_1 + x_2)(x_3 + x_4)(x_5 + x_6) \cdot ... \cdot (x_{n-1} + x_n)$$

איזו פולינום היא מייצגת? אולוהים יודע! היא ממש ממש ארוכה, ויש בה מספר אקספוננציאלי ב־n של מונומים.

4.2.3 הצגת הבעיה

בעיה: בהנתן נוסחא אריתמטית φ מעל $x_1, ..., x_n$, האם φ מייצגת את פולינום האפס?

מהדוגמא הקודמת, הדרך הלא נכונה לתקוף את הבעיה היא לפתוח את כל הסוגריים ולראות אם הכל מצטמצם - זה כאמור יקח זמן אקספוננציאלי, ועל כן זו לא הדרך.
אם כך, הכיצד?

הערה 4.23 נשים לב כי הבעיה הזו בעצם מייצגת גם את הבעיה, האם שתי נוסחאות המייצגות פולינומים זהות? אז פשוט מפחיתים אחד מהשניה ושואלים אם זהו פולינום האפס.

נגדיר את הפולינום שאנו מחפשים לעשות את הפעולה הבאה:
כאשר $\varphi \equiv 0$, יחזיר "yes", כאשר $\varphi \neq 0$, יחזיר "no".

הגישה לפתרון היא די פשוטה ודומה למה שעשינו קודם:

1. נבחר השמה $\underline{\alpha}$ רנדומית.

2. נחשב את $\varphi(\underline{\alpha})$.
3. נחזיר "$\varphi \equiv 0$" אמ"מ $\varphi(\underline{\alpha}) = 0$, אחרת "$\varphi \neq 0$".

נראה כיצד נפרמל את הגישה:
תחילה, כיצד נחשב את $\varphi(\underline{\alpha})$?
אבחנה: יש אלגוריתם פולינומי אשר בהנתן נוסחה אריתמטית φ במשתנים $x_1, ..., x_n$, ובהנתן השמה $\underline{\alpha}$ עבור משתנים אלה, מחשב את $\varphi(\underline{\alpha})$.
זוהי לא אבחנה טריוויאלית לחלוטין, אך היא נכונה.
נטען באינדוקציה כי מספר פעולות החשבון שהאלגוריתם יצטרך לעשות הוא לינארי באורך הנוסחא. זה נובע מההגדרה של נוסחאות - שתי האפשרויות הראשונות טריוויאליות, והן מקרה הבסיס. אם $\varphi = \varphi_1 \cdot \varphi_2$" לחישוב φ_1 מההנחה צריך $|\varphi_1|$, לחישוב φ_2 צריך $|\varphi_2|$, צריך עוד פעולה אחת לכפול, לכן צריך לא יותר מ-$|\varphi_1| + |\varphi_2| + 1$ פעולות, לכן לינארי באורך הקלט. מצד שני, כמה גדולים המספרים שעליהם צריך לעשות חישוב, נניח מבחינת כמות הביטים? כל מספר שמופיע ב-$\underline{\alpha}$ הוא לא יותר מאורך הקלט. מצד שני, יכול להיות שאנחנו כופלים מספרים, ואז כאשר כופלים אפשר לקבל אורך הרבה יותר גדול[40].
מהו אורך המספר הכי גדול שיכול להופיע תוך כדי החישוב? נסמן ב-m את אורך הקלט. כאשר כופלים מספרים באורך m, יהיו m^2 ביטים בתשובה. לכן, אורך הביטים בכל מספר לא יותר מאורך הקלט בריבוע. על כן, בזמן פולינומי אפשר לחשב את כל הנוסחא, כלומר האבחנה נכונה.
אזי, את השלב השני בגישה ניתן לבצע, אם מספר הביטים בכל מספר בהצבה הוא פולינומי במספר הביטים בקלט.

האם אפשר להגיד משהו על הדרגה של $\varphi(\underline{\alpha})$?
אבחנה 2:

$$deg(\varphi) \leq |\varphi|$$

אם יש נוסחא בהרבה משתנים, והיא לכל היותר מדרגה $|\varphi| = k$, על כמה השמות לכל היותר היא יכולה להתאפס? התשובה - אינסוף!
למשל:

$$\varphi : x_1 - x_2$$

כלומר, השיקול שעשינו קודם לא עובד. מה נעשה?

משפט שוורץ זיפל

משפט 4.24 יהי $p(x_1, ..., x_n)$ פולינום שאינו פולינום האפס מדרגה $\geq k$, ותהי H קבוצה לא ריקה של מספרים ממשיים. תהי $\alpha_1, ..., \alpha_n$ השמה מקרית הנבחרת באופן רנדומי מהמספרים בקבוצה H. אזי:

$$Pr_{\alpha_1,...,\alpha_n \in H}[p(\alpha_1, ..., \alpha_n) = 0] \leq \frac{k}{|H|}$$

ההוכחה באינדוקציה על מספר המשתנים. נשים לב כי אם מספר המשתנים כבר אחד, כבר אמרנו את זה קודם לכן.
כעת, אנו מוכנים לנסח את האלגוריתם:

אלגוריתם 1 Identity Test

בהנתן נוסחא φ במשתנים $x_1, ..., x_n$, האלגוריתם יפעל כדלהלן:

- יבחר לכל i, $\alpha_i \in \{1, ..., 2 \cdot |\varphi|\}$.
- יעריך את $\varphi(\alpha_1, ..., \alpha_n)$.
- אם יצא 0, יחזיר "$\varphi \equiv 0$", אחרת "$\varphi \neq 0$".

נשים לב כי האלגוריתם אכן פועל בזמן פולינומי, מהאבחנות הקודמות.
אבחנה 3: אם $\varphi \equiv 0$, האלגוריתם תמיד יחזיר "$\varphi \equiv 0$", ואם $\varphi \neq 0$, האלגוריתם יחזיר "$\varphi \neq 0$" בהסתברות $\leq \frac{1}{2}$.
בשביל הסתברות טעות יותר נמוכה, אפשר להגדיל את הקבוצה, או לחזור על האלגוריתם.

[40] כמובן שזה גם יכול לקרות עבור חיבור, אבל זה, כדברי המרצה, "קטן עלי", או לפחות קטן על המקרה של כפל.

4.3 Amplification

הגברה, אבל לא באמת... המטרה - הקטנת הטעות של האלגוריתם.
יהי M אלגוריתם רנדומי (RP) לשפה L. אם נרצה להקטין את הטעות, אפשר להגדיר מכונה חדשה M', אשר על קלט $x \in \{0,1\}^*$ תחזור על M $|x|$ פעמים. אזי:

$$\forall x \ Pr\left[M' \text{is wrong on } x\right] < \frac{1}{2^{|x|}}$$
$$\Rightarrow Pr_{r,x\in\{0,1\}^n}\left[M' \text{is wrong on } x\right] < \frac{1}{2^n}$$

מסקנה 4.25

$$\forall n \ \exists r \ s.t. \ \forall x \in \{0,1\}^n \ \ M' \text{answers right on } x \text{with the advice } r$$

דיברנו על RP - תזכורת:
$L \in RP$ אמ"מ יש מכונת טיורינג רנדומית M הרצה בזמן פולינומי כך ש:

- $Pr_r\left[M \text{accepts } x \text{with the advice } r\right] \geq \frac{1}{2} \Leftarrow x \in L$.
- $Pr_r\left[M \text{accepts } x \text{on } r\right] = 0 \Leftarrow x \notin L$.

טענה 4.26 בהנתן M כזו, ניתן לבנות מ"ט רנדומית M' הרצה בזמן פולינומי ומקיימת:

- $Pr_r\left[M' \text{accepts } x \text{on } r\right] = 0 \Leftarrow x \notin L$.
- $Pr_r\left[M \text{accepts } x \text{on r}\right] > 1 - \left(\frac{1}{|\Sigma|}\right)^{|x|} \Leftarrow x \in L$.

הרעיון - M' תריץ את M כאורך x פעמים, הסיכוי שהיא תטעה בכל הפעמים האלו היא $\frac{1}{2^{|x|}}$, ואם תריץ $|x| \cdot \log|\Sigma|$, תתקבל ההסתברות הנ"ל.
יישום זה כאמור נקרא הגברה - amplification.

טענה 4.27 יהא $n \in \mathbb{N}$. אז קיימת עצה r עבורה:

$$\forall x \ \ (|x| = n): \ \ (x \in L) \Leftrightarrow (M' \text{accepts } x \text{on } r)$$

הוכחה: נסמן:

$$1(x,r) = \begin{cases} 1 & M' \text{is wrong on } x \text{with } r \\ 0 & \text{otherwise} \end{cases}$$

אזי:

$$\forall x \ \ (|x| = n): \ \ \mathbb{E}_r\left[1(x,r)\right] = Pr_r\left[M \text{is wrong on } x \text{with } r\right] \overset{\text{prev. claim}}{<} \left(\frac{1}{|\Sigma|}\right)^n$$
$$\Rightarrow \mathbb{E}_r\left[\sum_{x\in\Sigma^n} 1(x,r)\right] \overset{\text{linearity of } \mathbb{E}}{<} 1$$

מכיוון ו"הממוצע של הרבה דברים קטן מאחד, אז לפחות אחד מהם קטן מאחד". כלומר:

$$\Rightarrow \exists r \ s.t. \sum_{x\in\Sigma^n} 1(x,r) < 1$$
$$\Rightarrow \exists r \ s.t. \ \forall x, \ \ (|x| = n) \Rightarrow 1(x,r) = 0 \Leftrightarrow M' \text{is not wrong on } x \text{with } r$$

שכן הפונקציה המציינת $1(x,r)$ מקבלת את הערכים $\{0,1\}$ בלבד, לכן אם הסכום קטן מאחד, בהכרח הוא 0. ■

מסקנה 4.28 תהי $L \in RP$. אזי, יש מכונת טיורינג M עם עצה שרצה בזמן פולינומי כך שקיימת עצה r_0 שעבורה $M \Leftrightarrow x \in L$ מקבלת את x על r_0.

זה מאוד דומה להגיד ש־$RP \subseteq P$. אבל, המכונה הפולינומית הזו רק מכריעה אם קובעים לה את העצה. העצה הזו היא מחרוזת אינסופית, אז צריך לקחת את המכונה, להוסיף מחרוזת אינסופית, וכאן הבעיה (וכמובן שלא כל מחרוזת אינסופית תעבוד, אלא רק אחת מיוחדת).

נוכיח את המסקנה. **הוכחה:** r_0 יהיה שרשור העצות מהטענה הקודמת עבור כל אורכי הקלטים.

תחילה, M' תבדוק מהו האורך של הקלט x. לאחר מכן, היא "תרוץ" על r_0 עד שתמצא את העצה עבור קלט באורך $|x|$, ואז תרוץ על העצה הזו.

כל מהלך הוא פולינומי, על כן M' רצה בזמן פולינומי, כנדרש. ■

הטענה הזו מראה כי הרנדומיות מוסיפה לנו כח, אבל לא הרבה. כדי להגיד מה בדיוק הוכחנו פה, צריך הגדרה:

הגדרה 4.29 שפה L היא ב־$P/poly$ אם יש מ"ט עם עצה הרצה בזמן פולינומי כך שקיימת r_0 המקיימת $M \Leftrightarrow (x \in L)$ מקבלת את x עם r_0.

$P - polynomial\ time$, $poly$ - עצה באורך פולינומי.

אבל רגע - העצה שכתבנו היא באורך אינסופי. אבל המכונה רצה בזמן פולינומי, ולכן משתמשת במספר פולינומי של אותיות מתוך העצה.

אנו יודעים כי $P/poly \not\subseteq P$, שכן מחלקה זו מכילה שפות שאינן כריעות בכלל.

מטעמי ספירה, קיימת שפה לא כריעה מעל א"ב אונרי, יתרה מזו, אנו מכירים כאלו - A_{TM} בקידוד אונרי, למשל. מדוע? לכל אורך n, קיימת רק מילה אחד באורך n בקידוד זה, לכן r_0 יכולה פשוט להכיל ביט שיגיד אם המילה הזו בשפה או לא. $P/poly$ לא למבחן!

על המבחן

20/6/2012

לגבי החומר - החומר הוא הכל!

כלומר, כל התרגילים מצופים להיות פתירים בזמן מבחן, כל המשפטים שהוכחו או בשיעור או בתרגול יש לדעת לשחזר את המשפט ואת הוכחתו, ויש לדעת לדקלם את כל ההגדרות.

זה לא מספיק בכלל "לזכור בערך את המילים שמופיעות שם ולכתוב ואתן באיזשהוא סדר על הדף". צריך באמת להפנים באופן עמוק את ההגדרות, הטענות וההוכחות.

המבחן בחומר סדור, לא יהיו הארכות. אם כולם לא הצליחו לפתור שאלה, רב הסיכויים שיתחשבו בזה בעת מתן הציונים. זה גם אומר שאם נתקעים בסעיף שלא יודעים לפתור - להמשיך הלאה, ואם יש זמן לחזור עליו.

מבנה הבחינה:

הבחינה עדיין לא רשומה, ולכן כל הרשום מטה בערבון מוגבל:

המבנה יהיה פחות או יותר כך[41]:

- שאלת הוכחה אחת לפחות - טענה שהוכחה בכיתה, חלק מטענה שהוכחה בכיתה, או חלק מטענה שניתנה בכיתה תחת ניסוח שונה (הפעלת עקרון מתוך הוכחה שאנו מכירים).
- עוד שאלות.
- (אולי) שאלות כן\לא\בחירה מרובה - לא להתבלבל, שאלות כאלה יכולות לדרוש מחשבה. שאלות כאלו אם יהיו ידרשו נימוק בנוסף לתשובה (כדי להמנע ממתן ניקוד גבוה על ניחוש מוצלח) - נימוק, לא הוכחה ריגורוזית.

לא תהיה בחירה.

בשני הסעיפים הראשונים - שאלות אלו בדר"כ בסעיפים הקשורים אחד לשני, וכך אפשר "לאסוף נקודות" למשל על הגדרות וניסוחים של משפטים.

בחלק מהשאלות, בעיקר בחלק השני, יכול להיות שסעיף אחד יהיה ניסוח טענה, והשני הוכחה של משהו. השאלות לא מקריות, אם לא רואים קשר שווה להשקיע זמן ולהבין את הקשר בין הסעיפים. כמו כן, לפעמים יש דרגות משתנות של קושי בין הסעיפים (סדר עולה). שווה לעשות קודם את הסעיפים הראשונים בכל שאלה, ואז להמשיך לסעיפים הקשים יותר.

כשעוברים על מבחנים משנים קודמות, לפני פאניקה יש לחשוב אם למדנו השנה את החומר. כמו כן השנה כיסינו חומר שאולי לא כוסה בשנים הקודמות.

חשוב - להבין את "הפואנטה" של השאלה, להשקיע זמן בכתיבה על מה ששואלים ולא על דברים מסביב.

עוד דבר מאוד חשוב - לקרוא את השאלות בבחינה!

ועוד - אם יש זמן, לקרוא את התשובות שכתבנו.

[41] "מתוך כל האי וודאות שבחיים, זו אחת הקטנות שבהן", דברי המרצה.

אם רוצים להשתמש בטענות קודמות ולא בטוחים אם מותר - לשאול בזמן מבחן את המתרגלים. עם זאת, יש הרבה דברים שאפשר להפעיל בהם שיקול דעת עצמאי[42].

על הניקוד יהיה ניקוד לכל שאלה, לא יהיה ניקוד לכל סעיף, זה מותר לפי התקנון.

איך המרצה היה לומד לבחינה לעבור על כל הרשומות מכל השיעורים.
יעזור לעשות רשימת הגדרות וטענות.
לעבור על כל התרגילים ולוודא שיודעים לפתור את הכל. לפחות חלק מהם - לכתוב ממש את הפתרון, לוודא שבאמת יודעים, ולהשוות ולפתרונות הקורס.
כשעוברים על התרגילים - אפשר לחשוב עליו בטריגר לחזור אחורה. אם לא יודעים לפתור משהו - לפתוח את החלק הרלוונטי. ולקורא שוב.
לעבור על בחינות משנים קודמות. לפחות את חלקן לפתור בזמן אמת ולבדוק מה היו הטעויות.
עוד דבר חשוב - לנוח, לאכול ולישון[43].

[42] אם הוכחנו בכיתה ש-$A_{TM} \notin R$, כדי להוכיח אותו אי אפשר להשתמש בזה שראינו את זה בכיתה.
[43] "החיים האמיתיים מאוד קשים, העולם המערבי שוקע", דברי המרצה.

חלק II
תרגולים

תרגול 2

The Pumping Lemma

למה 4.30 תהי $L \in REG$. אזי קיים קבוע $p > 0$ כך שעבור כל $w \in L$ המקיימת $|w| > p$, קיימים $x, y, z \in \Sigma^*$ כך ש־$w = xyz$ וכן:

1. $|y| > 0$.
2. $|xy| \leq p$.
3. לכל $i \ni \mathbb{N} \cup \{0\}$, $xy^i z \in L$.

אפשר להשתמש בה על מנת להוכיח ששפה אינה רגולרית באופן הבא:
מניחים בשלילה שהשפה רגולרית, ואז מוצאים מילה בשפה, חלוקה שלה לשלושה חלקים ו־i שיסתור את תנאי 3.

תרגיל 3

Myhill Nerode

28/03/2012

תהי $L \subseteq \Sigma^*$, Σ סופי. הגדרנו את היחס $\sim_L \Sigma^* \times \Sigma^*$:

$$\forall x, y \in \Sigma^* \; x \sim_L y \Leftrightarrow \forall z \in \Sigma^* \; x \cdot z \in L \Leftrightarrow y \cdot z \in L$$

כמו כן הגדרנו מחלקת שקילות:

$$\forall w \in \Sigma^* \;\; [w] = \{x | x \sim_L w\}$$

וכן ראינו משפט:

משפט 4.31 $L \in REG$ אמ"מ מס' מחלקות השקילות של $\sim_L$ הוא סופי.

היופי במשפט - הוא אמ"מ, ומאפיין לנו בדיוק את קבוצת השפות הרגולריות.

דוגמאות

1.

$$L = \left\{a^k | \text{k is not a power of 2}\right\}$$

בתרגיל רואים שהשפה המשלימה לשפה זו אינה רגולרית. על כן, גם השפה הזו אינה רגולרית.
נראה שיש אינסוף מחלקות שקילות $\sim_L$. לכל $n < m$, נתבונן במילים a^{2^n}, a^{2^m}, ונטען ש־$a^{2^m} \nsim_L a^{2^n}$. כדי לעשות זאת, נמצא זנב מפריד - נתבונן בזנב a^{2^n}:

$$a^{2^n} \cdot a^{2^n} = a^{2 \cdot 2^n} = a^{2^{n+1}} \notin L$$
$$a^{2^m} \cdot a^{2^n} = a^{2^m + 2^n} \overset{(*)}{\in} L$$
$$(*)\, 2^m + 2^n = 2^{m-n} \cdot 2^n + 2^n = 2^n \left(1 + 2^{m-n}\right)$$

באותה צורה אפשר גם להוכיח כי $a^{2^n} \nsim_L a^{2^{n+1}} \nsim_L a^{2^{n+2}}$ אבל זה לא יעזור - היחס אינו טרנסיטיבי. אולי מדובר רק בשתי מחלקות שקילות? לכן יש להוכיח את מה שהוכחנו מעלה, וזה מספיק.

2.

$$L = \{w | \text{w ends with a}\}$$

זוהי שפה רגולרית. נחשב את מחלקות השקילות. נטען שיש שתי מחלקות שקילות - מילים שמסתיימות ב־a, ומילים שלא מסתיימות ב־a.
תחילה נראה כי אם x, y מסתיימות ב־a, אז $x \sim_L y$:

$$\forall z \in \Sigma^* \; x \cdot z \in L \Leftrightarrow z \text{ ends with } a \text{or } z{=}\varepsilon \; \Leftrightarrow y \cdot z \in L$$

שנית, אם x, y לא מסתיימות ב־a, נראה כי $x \sim_L y$:

$$\forall z \in \Sigma^* \; x \cdot z \in L \Leftrightarrow z \text{ends with } a \Leftrightarrow y \cdot z \in L$$

איחוד של שתי המחלקות הללו מהווה את כל Σ^*, ועל כן השפה רגולרית. לסיום, נראה כי יש בדיוק שתי מחלקות שקילות, כלומר ששתי המחלקות מעלה נפרדות זו מזו: אם x מסתיימת ב־a ו־y לא:

$$x \cdot \varepsilon \in L, \; y \cdot \varepsilon \notin L, \; x \nsim_L y$$

3. נביט בשתי מחלקות שקילות:

$$\{u | u \text{ends with } a\}, \; \{u | u \text{doesn't end with } a\}$$

נמצא אוטומט מתאים לשפה: נקיף את שתי המחלקות, ונכתוב את המעברים בהתאם לא"ב. אי אפשר למצוא אוטומט עם פחות משני מצבים (או, פחות ממספר מחלקות השקילות) - זה יישום של הוכחת המשפט של מייהיל נרוד.

מינימיזציה של DFA

בהנתן DFA A, האם ניתן למצוא DFA שקול מינימלי?
אם אנחנו יודעים את מחלקות שקילות מייהיל נרוד, כן, לפי המשפט. ואם לא? אפשר!
הרעיון - ניקח את האוטומט שלנו ו"נמעך" מצבים ביחד, כך שלא יהיה אכפת לנו מה נקרא, נדע לאן להמשיך. זוהי בעצם דרך למציאת מחלקות מייהיל־נרוד.
עבור A, נגדיר $\sim_A \; Q \times Q$ בצורה הבאה:

$$q \in \sim_A s \Leftrightarrow \forall z \; \delta^*(q, z) \in F \Leftrightarrow \delta^*(s, z) \in F$$

נשים לב כי אנו לא דורשים שיוויון ממש, כי אנחנו רק רוצים לוודא ששתי המילים מתקבלות, ולאו דווקא שיגיעו לאותו מצב מקבל. כמו כן, נשים לב שכאן אנו מגדירים שקילות בין מצבים, ולא בין מילים.
נניח שידועות לנו מחלקות השקילות של $\sim_A$. נטען שאם נשלב את כל המצבים באותה מחלקת שקילות למצב אחד, נקבל אוטומט שקול.
ציור של אוטומט
באוטומט בציור, $1, 2$ הם שקולים, וכן $3, 4$ הם שקולים. אוטומט מינימלי לשפה הזו יכל רק שני מצבים:
ציור של אוטומט מינימלי
אם כך, כל מה שנשאר הוא למצוא את מחלקות השקילות של A. יהי A' האוטומט לאחר השילוב הנ"ל, כאשר:

$$\delta'([q], \sigma) = [\delta(q, \sigma)]$$

נשאר לקורא השקדן להוכיח כי אין כאן בעיה בהגדרה.

טענה 4.32

1. $L(A') = L(A)$.
2. A' מינימלי.

הוכחה: נוכיח את 2 - נוכיח שב־A' יש לכל היותר מספר מצבים כמספר מחלקות $\sim_{L(A)}$, ומהמשפט מתחילת השיעור זה יספיק:

$$x \sim_{L(A)} y \ iff \ \forall z \ \ (xz \in L \Leftrightarrow yz \in L) \ \ iff \ \delta^* (q_0, xz) \in F \Leftrightarrow \delta^* (q_0, yz)$$
$$iff \ \delta^* (\delta^* (q_0, x), z) \in F \Leftrightarrow \delta^* (\delta^* (q_0, y), z) \in F \ iff \ \delta^* (q_0, x) \sim_A \delta^* (q_0, y)$$

כעת נותר למצוא את מחלקות השקילות של המצבים. הבעיה בבדיקה ישירות היא שנצטרך לבדוק את כל הזנבות ב־Σ^*, ויש אינסוף כאלו.
נגדיר סדרה של יחסים $\sim_i$ בצורה הבאה:

$$q \sim_i s \Leftrightarrow \forall z, \ |z| \leq i \ \delta^* (q, z) \in F \Leftrightarrow \delta^* (s, z) \in F$$

נתחיל ב־$i = 0$ - שם כל המצבים המקבלים יהיו במחלקת שקילות אחת, והלא מקבלים בשניה.
כדי לחשב את $\sim_{i+1}$ מתוך $\sim_i$, נעבור על כל זוג מצבים $q \sim_i s$ ונבדוק לכל $\sigma \in \Sigma$ האם:

$$\delta (q, \sigma) \sim_i \delta (s, \sigma)$$

אם מצאנו אות עבורה שני מצבים אינם שקולים עבור i כלשהוא, אזי הם לא שקולים עבור $i+1$. נמשיך עד שנמצא i כך ש:$\sim_i = \sim_{i+1}$, ואז נדע שהגענו ליחס הסופי $\sim_A$, שכן אז לא משנה כמה אותיות נוסיף, השקילות נשמרת. ■

ביטויים רגולריים - Regex

כח ההבעה של אוטומטים הוא מוגבל - אם כל מה שידע המחשב לעשות היה ליישם אוטומטים, השימושיות היתה מאוד מוגבלת. למה זה כן טוב? אם אנחנו רוצים "להאכיל" אלגוריתם בשפה, נראה מאוד יעיל להשתמש באוטומט שלו.
יש שיטות ייצוג אחרות של שפות רגולריות, שנוחות יותר לשימוש. אחת מהן - ביטויים רגולריים.
למשל:

$$((a \cup b) \ \ baa \ c^*) \bigcup (bb \cup ac)^*$$

הגדרה 4.33 ביטוי רגולרי r הוא אחד מהבאים:

- ϕ.
- ε.
- a כאשר $a \in \Sigma$.
- $s \cup t$ עבור ב"ר s, t.
- $s \cdot t$ עבור ב"ר s, t.
- s^* עבור ב"ר s.

ונכתוב:

$$r := \phi | \varepsilon | a | r \cup r | r \cdot r | r^*$$

הגדרה 4.34 עבור $L \subseteq \Sigma^*$ ועבור $k \in \mathbb{N} \cup \{0\}$:

$$L^k = \overbrace{L \cdot L \cdot ... \cdot L}^{k \ times} = \left\{ w^1 \cdot w^2 \cdot ... \cdot w^k | \forall i \ w^i \in L \right\}$$
$$L^0 = \{\varepsilon\}, \ L^* = \bigcup_{k \in \mathbb{N} \cup \{0\}} L^k$$

הגדרה 4.35 בהנתן ב"ר נגדיר את $L(r)$ להיות:

- $L(\phi) = \phi$.
- $L(\varepsilon) = \{\varepsilon\}$.
- $L(a) = \{a\}$.
- $L(r \cup s) = L(r) \cup L(s)$.
- $L(r \cdot s) = L(r) \cdot L(s)$.
- $L(r^*) = (L(r))^*$.

דוגמא

$$(a \cup b)^* bb (a \cup b)^*$$

השפה של הביטוי הזה - כל המילים שיש להן שני b באמצע - אפשר לבנות עץ ולראות את זה.

משפט 4.36 אמ"מ קיים ב"ר r כך ש־$L(r) = L$.

הוכחת המשפט - בשני כיוונים: נבנה מאוטומט ביטוי רגולרי, ובכיוון השני - לקחת ביטוי רגולרי ולבנות ממנו אוטומט.
את הכיוון הראשון נעשה בכיתה, השני עכשיו: **הוכחה:** $\Rightarrow$ נתון ב"ר r, נבנה ממנו NFA:
אם $r = \phi$, נבנה לא אוטומט עם מצב יחיד ולא מקבל.
אם $r = \{\varepsilon\}$, נבנה אוטומט עם מצב יחיד ומקבל.
אם $r = \{a\}$, נבנה אוטומט עם שני מצבים, הראשון לא מקבל, השני מקבל.
עבור ב"ר $r \cup s$, נחבר את שני האוטומטים של r, s.
עבור ב"ר מהצורה $r \cdot s$, פשוט נשתמש באוטומט השרשור כפי שכבר ראינו.
בתרגיל נראה כיצד ניתן לבנות אוטומט עבור ב"ר מהצורה r^*. ■

דוגמא

$$(a \cup b) \cdot b$$

ציור של אוטומט
אוטומט ל־a, אוטומט נפרד ל־b, אוטומט נוסף ל־b, ואז נרצה לחבר בין שני האוטומטים הראשונים לשלישי - נבטל את המצבים המקבלים משני האוטומטים הראשונים, ונחבר את האוטומטים ע"י מעבר מהמצב הראשון של אחד משני הראשונים למצב הראשון של השלישי, ונבטל את היות המצב הראשון של האוטומט השלישי מצב ראשון.

$2 - way - DFA$

כיצד מכלילים את המודל שלנו כדי שיהיה לו יותר כוח?
הצעה ראשונה - זיכרון, שכן לאוטומט אין זיכרון. על זה נדבר הרבה (מכונות טיורינג).
הצעה שניה - האפשרות ללכת אחורה. זה מה שעושים ב־$2 - way - DFA$.
נכיר את המושג טייפ - על הטייפ יש תאים. בתא הראשון והאחרון נשים דלימטר כדי שהאוטומט ידע מתי להתחיל ומתי להפסיק. תחילה, "הראש הקורא" נמצא בתא הראשון. כל צעד הוא מתקדם בהתאם לאות הבאה, וכן מוסיף כיוון - ימינה או שמאלה, כלומר $L \backslash R$, כאשר התקדמות היא ימינה, וחזרה אחורה - שמאלה.
לא ניתן הגדרה פורמלית למודל הזה, אבל נגדיר:

$$\delta : Q \times \Sigma \to Q \times \{L, R\}$$

אז נוכל להגדיר ריצה כרצף של קונפיגורציות:

$$\langle q_0, 0 \rangle \to \langle q_1, i \rangle$$

דוגמא - למה זה טוב, או, למה המודל הזה חזק מאוד

בהנתן:

$$L_k = \{w | \text{the k'th letter before last is a}\}$$

ראינו שקשה מאוד לבנות אוטומט שמזהה את השפה הזו - עם 2^k מצבים. עם זאת, במודל הזה, קל לאתר שפה זו עם $k-1$ מצבים:
אוטומט
ממשיך ימינה עד שמגיע לדילימטר, אז חוזר שמאלה k צעדים, אם בצעד הבא מקבל a, נגיע למצב מקבל.
עם זאת, אי אפשר להביע שפות שאינן רגולריות:

משפט 4.37 (רבין, סקוט '59) לכל $2-way-DFA$ יש NFA שקול.

$$A, B, (1,1), (3,6), (4,2), (1,1), (3,2), (2,2)$$

ההוכחה - ברשימות של התרגול.
בהמשך נוסיף את האפשרות של לכתוב על הטייפ, וזה מה שאפשר לעשות היום בעולם.

תרגול 4

מציאת $Regex$ בטקסט

18/4/2012

למשל:

$$((ab^*b)\cdot a^*)^* \cup ab \cup \_ = r$$

בעיה: בהנתן מילה x, נרצה לבדוק האם $x \in L(r)$.
פתרון: נבנה מ-r NFA שקול Ar.
נבנה DFA שקול Dr ונבדוק האם $w \in L(Dr)$.

בעיה: בהנתן טקסט w, האם יש ל-w תת-מחרוזת x כך ש-$x \in L(r)$.
פתרון: נעבור על כל תת-מחרוזת של w ונבדוק עבורה.
זה נראה מסובך, אבל זה בסה"כ $O(n^3)$. זה לא אסון, אבל נראה שאולי יש משהו יותר יעיל.
שיפור: אותו דבר רק עוצרים במצב מקבל - זוהי סיבוכיות של $O(n^2)$. אבל אנחנו רוצים יותר טוב!
שיפור נוסף: בנינו את Ar, נבנה ממנו את:
ציור!
מנחשים איפה מתחילים ה-infix ונבדוק אם מגיע למצב מקבל.
לפתרון זה זמן ריצה של $O(n)$!
פתרון אחר והרבה יותר פשוט:

$$(a \cup b)^* r (a \cup b)^*$$

להריץ את זה! לזה סיבוכיות של $O(n)$.

מכונת טיורינג

ראינו את ההתחלה של ההגדרה של מכונת טיורינג בכיתה.
לא הוספנו שבשלב מסויים מגיעים למצב q_{acc} או q_{rej} - אם לא מגיעים לאחד משני המצבים האלו אומרים שהמכונה "נתקעת". נגדיר זאת פורמלית בכיתה.
כאן לא נדבר על ההגדרה הפורמלית, אולם נזכיר את הרכיבים:

- Q - קב' מצבים.
- Σ - א"ב קלט.
- Γ - א"ב סרט, המקיים $\Sigma \subseteq \Gamma$, $\_ \in \Gamma \backslash \Sigma$.

- δ - פונקצית מעברים:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$$

- q_0 - מצב התחלתי.
- q_{acc}, q_{rej}.

דוגמא נראה שמכונת טיורינג יכולה לזהות את השפה:

$$L = \left\{ w \in \{a, b\}^* \mid \#_a(w) \leq \#_b(w) \right\}$$

תיאור $high - level$:

1. נסרוק את w ונחפש a. אם מצאנו - נמחק את ה-a ונחפש b. אם לא מצאנו a, נקבל.
2. בחיפוש b - אם מצאנו b נמחק ונחזור a, אם לא מצאנו b, נדחה.

זהו תיאור אלגוריתמי. איך נעביר את זה למכונת טיורינג?
תיאור $low - level$:

$$\Gamma = \{x, \_\} \cup \Sigma$$

כאשר x שמים לאחר מחיקה.

- $Search(a)$:
  - בקריאת b, x נלך ימינה.
  - בקריאת _ נקבל.
  - בקריאת a, נכתוב x, ונעבור ל-$Found(a)$.
- $Found(a)$:
  - בקריאת כל דבר מלבד _, נלך ימינה.
  - בקריאת _ נעבור ל-$Search(b)$.
- $Search(b)$:
  - נלך שמאלה עד שנמצא b, אם מצאנו נכתוב x ונעבור ל-$Found(b)$.
  - אחרת נדחה.
- $Found(b)$:
  - נלך שמאלה עד _, ונעבור ל-$Search(a)$.

הוכחת הנכונות של דבר כזה - אינה פורמלית, ממש כמו התיאור מעלה.
פעם אחת נכתוב תיאור ממש פורמלי של המכונה:

תיאור פורמלי של מכונת טיורינג

$$Q = \{s_a, s_b, f_a, f_b, q_{acc}, q_{rej}\}$$
$$\Gamma = \{X, \_, a, b\}$$
$$q_0 = s_a$$

את פונקציית המעברים נתאר בטבלה:

δ	a	b	X	$\_$
s_a	$\langle f_a, X, R\rangle$	$\langle s_a, b, R\rangle$	$\langle S_a, X, R\rangle$	$\langle q_{acc}, \_, L\rangle$
f_a	$\langle f_a, a, R\rangle$	$\langle f_a, b, R\rangle$	$\langle f_a, X, R\rangle$	$\langle s_b, \_, L\rangle$
s_b	$\langle s_b, a, L\rangle$	$\langle f_a, X, L\rangle$	$\langle s_b, X, L\rangle$	$\langle q_{rej}, b, R\rangle$
f_b	$\langle f_b, a, L\rangle$	$\langle f_b, b, L\rangle$	$\langle f_b, X, L\rangle$	$\langle s_a, \_, R\rangle$

- ריצה של מכונת טיורינג על מילה היא סדרה סופית של קונפיגורציות עוקבות.
- קונפיגורציה של מכונת טיורינג: מה כתוב על הסרט, והיכן נמצא הראש הקורא, ובאיזה מצב אנחנו. למשל:

$$abb\overset{b}{q}aba$$

כאשר ה־q הוא מצב.

למשל, נראה את ריצת מכונת טיורינג על המילה $babba$:

$$\overset{b}{S_a}abba$$
$$b\overset{a}{S_a}bba$$
$$bx\overset{b}{F_a}ba$$
$$bxb\overset{b}{F_a}a$$
$$bxbb\overset{a}{F_a}$$
$$bxbbx\overline{F}_a$$
$$bxbb\overset{a}{S_a}$$
$$\vdots$$
$$bxxxx\_\overline{q_{acc}}$$

מכונת טיורינג עם סרט חד צדדי מכונת טיורינג יכולה למסמלץ (מלשון לעשות סימולציה) את המודל עם סרט החסום משמאל ("חד צדדית") ע"י כתיבת סימון מיוחד $ משמאל למילה, והוספת המעבר:

$$q \overset{\$\to\$,R}{\to}$$

מצד שני, מכונת טיורינג חד צדדית (חסומה מצד אחד) יכולה לסמלץ מכונת טיורינג ־ לוקחים שתי מכונות חד צדדיות, כשרוצים לעבור לצד החסום פשוט עוברים מכונה (רואים את זה על פי סימן מיוחד $).
זה אמור להיות רשום בסיכום התרגול.

חישובים בעזרת מכונת טיורינג

המודל של מכונת טיורינג לא בנוי לתת תשובות אחרות מלבד "כן" ו"לא" (מקבל או לא מקבל). עם זאת, אפשר להשתמש במכונת טיורינג לחישוב פונקציות!

למשל, נחשב את הפונקציה:

$$f(n) = n + 1$$

מה הקלט למכונה? נשאל איך נייצג את n - דצימלית? (לא כדאי) אונרית? בינארית? שתי האופציות האחרונות מאוד שונות זו מזו.

נעשה זאת בבינארית.

מה תהיה פונקציית המעברים?

δ	0	1	$\_$
q_r	$q_r, 0, R$	$q_r, 1, R$	$q_f, \_, L$
q_f	$q_{acc}, 1, L$	$q_f, 0, L$	$q_{acc}, 1, L$

מדוע? הולכים עד לסוף הקלט n ימינה, ואז מתחילים לבדוק -

אם הספרה שקראנו הוא 0, מספיק להחליף אותו ב-1 כדי להגדיל את המספר ב-1, על כן לאחר השינוי הזה על הסרט כתוב $n + 1$, ואז נקבל. אחרת אם קראנו 1, נחליף ל-0, ואבל עכשיו צריך לשנות גם בהמשך (חיבור של מספרים ביצוגם הבינארי כפי שראינו כבר בקורסים קודמים), אז נמשיך לספרה הבא משמאל - אם קראנו 0 נחליף ב-1 וסיימנו, אחרת נמשיך בתהליך הזה. אם לא נמצא 0 בדרך, נגיע לספרה הימנית ביותר, נשים במקומה 1, ונסיים.

תרגול 5

מודלים שקולים למכונת טיורינג

"והפעם, בפינתנו "מה שקול למכונת טיורינג"":

מכונת טיורינג עם שני סרטים

הגדרה 4.38 $2-TM$ היא מכונת טיורינג עם שני סרטי עבודה, ושני ראשים.
פונקציית המעברים היא:

$$\delta : Q \times \Gamma \times \Gamma \to Q \times \Gamma^2 \times \{R, L\}^2$$

בתחילת הריצה של $2-TM$, על סרט אחד כתוב הקלט, וסרט 2 ריק.

דוגמא: *ציור*

טענה 4.39 $2-TM$ שקול ל-TM.

נשאל, מה בכלל הכוונה בשקילות?

הגדרה 4.40 מכונת טיורינג M שקולה ל-$2-TM$ M' אם לכל $w \in \Sigma^*$:

- M מקבלת את w אמ"מ M' מקבלת את w.
- M דוחה את w אמ"מ M' דוחה את w.

מכיוון וכיוון ואחד ברור (אם מוסיפים סרט לא פוגעים ביכולות של המכונה), נוכיח רק את הצד השני:

טענה 4.41 לכל $2-TM$ יש TM שקול.

הוכחה: תהי $2-TM$ M. נגדיר את TM M' באופן הבא:
הא"ב של M' יהיה:

$$\Gamma' = \Gamma \times \Gamma \times \{0, 1\} \times \{0, 1\}$$

1. M' רצה ימינה, מחליפה כל אות σ באות $\begin{pmatrix} \sigma \\ \_ \\ 0 \\ 0 \end{pmatrix}$. לאחר מכן נחזור שמאלה ונעדכן את האות השמאלית ל-$\begin{pmatrix} \sigma \\ \_ \\ 1 \\ 1 \end{pmatrix}$.

2. M' מקודדת במצבים את המצב של M:

(א) נמצא את ראש 1 ונזכור את האות מעליו.

(ב) נמצא את ראש 2 ונזכור את האות שמעליו.

(ג) נמצא שוב את ראש 1, ונעדכן לפי δ.

(ד) נמצא שוב את ראש 2, ונעדכן לפי δ.

אם בשלב כלשהו M עוברת למצב מקבל\דוחה, נקבל\נדחה בהתאם.

■

איך משתנה זמן הריצה?

בהנתן TM שרץ m צעדים, כמה זמן תרוץ $2-TM$ מסמלצת?
עבור כל צעד אחד של M, M' תבצע $O(n)$ פעולות, כאשר n הוא אורך הסרט. לכן, נצטרך סה"כ $O(n \cdot m)$, ומכיוון שמכונה שרצה m צעדים יכולה לכתוב לכל היותר m תאים, $n = O(m)$, ולכן בסה"כ $O(m^2)$.

ריצות במקביל

נזכור שברגע שאנחנו יודעים קונפיגורציה במדויק, אנחנו יודעים כיצד להמשיך הלאה.

טענה 4.42 אם $L_1, L_2 \in RE$, אז $L_1 \cdot L_2 \in RE$.

הבעיה: ב־RE אם נתחיל לעבור על פירוקים אפשריים, יתכן שנתקע, בעוד שבהמשך אולי יש פירוק חוקי שלא הגענו אליו, לכן אסור לדחות. מה נעשה? **הוכחה:** יהיו M_1, M_2 מכונות המזהות את L_1, L_2 בהתאמה. נבנה מכונה M_3 כך ש:

$$L(M_3) = \{u\#v | u \in L_1,\ v \in L_2\}$$

M_3 פועלת כך: תריץ את M_1 על u, אם קיבלה, תריץ את M_2 על v, ותפעל בהתאם.
נשים לב כי M_3 לא בהכרח מכריעה.
כעת, נבנה את M כך:
ל־M שלושה סרטים: על הסרט הראשון כתוב הקלט, על הסרט השני ־ שני מונים (מונה אחד ל־i של האיטרציה, מונה שני שנוכל להוריד כדי לספור עבור כל פירוק), ועל השלישי ־ העבודה.
בכל איטרציה, M תעבור על כל פירוק של הקלט, ותריץ את M_3 על הפירוק (בסרט 3), למשך i צעדים, כאשר i הוא המונה של האיטרציה בסרט 2.
אם אף פירוק לא מתקבל, M תגדיל את המונה בסרט 2.
כך, המונים למעשה מגביל לנו את מספר הריצות ־ אם המילה מתקבל, היא תתקבל אחרי מספר סופי כלשהוא של צעדים, על כן נמצא את הפירוק. ■

המעבר מבעיות הכרעה לשפות

נתון NFA ומילה w, האם $w \in L(A)$? זוהי בעיית הכרעה. איך נתאר אותה כשפה?

$$L = \{\langle \mathcal{A}, w \rangle | \mathcal{A} \text{ is an NFA}, w \in \Sigma(\mathcal{A}),\ w \in L(\mathcal{A})\}$$

נקבע א"ב $\Sigma = \{0, 1, \#\}$, ונקודד את $\mathcal{A}$ באופן "סטנדרטי":

$$\overbrace{\langle\sigma_1\rangle \# \langle\sigma_2\rangle \#...\#}^{\langle\Sigma\rangle} \langle Q\rangle \#\# \langle\delta\rangle \#\#$$

נשים לב שלמכונת טיורינג קל מאוד לוודא שקלט נתון הוא אכן קידוד חוקי (כולנו מימשנו פרסרים בשלב כזה או אחר ־ מגעיל, אבל לא מסובך).

איך קידוד משפיע?

נביט ב:

$$L = \{\langle p \rangle | p \text{ is primary}\}$$

זמן ריצה של אלגוריתם נאיבי: $O(p)$. אבל, זמן הריצה הזה הוא לא תלוי בקלט.
נשאל, האם $O(|\langle p \rangle|)$ הוא זמן ריצה נכון? זה תלוי בקידוד! זה נכון עבור קידוד אונרי.
אם מקודדים בבינארי, זמן הריצה הופך ל־$O\left(2^{|\langle p \rangle|}\right)$. זהו הבדל רציני מהתוצאה הקודמת! נראה את החשיבות של הדבר הזה כאשר נדבר על מחלקות סיבוכיות, בהמשך הקורס.

משפט הרקורסיה $38'$ $[Kleene]$

טענה 4.43 קיימת מכונת טיורינג $SELF$, כך שבהנתן הקלט Σ, $SELF$ עוצרת כאשר על הסרט כתוב $\langle SELF \rangle$.

מה זה אומר? זוהי התנהגות וירוסית - מכונה שיודעת לעשות דבר כזה יכולה להריץ את עצמה, ואז למעשה היא תרוץ אינסוף פעמים.
$SELF$ תהיה מורכבת משתי מכונות A B, כך ש־A רצה, ולאחר מכן מעבירה את השליטה ל־B. איך עושים את זה?
פעולת A: הדפס $\langle B \rangle$.
כדי להגדיר את פעולת B, נזדקק לטענה הבאה:

טענה 4.44 קיימת מכונת טיורינג M, כך שבהנתן $w \in \Sigma^*$, M עוצרת כאשר על הסרט כתוב $\langle p_w \rangle$, כאשר p_w היא מכונה שמדפיסה את w.

אנחנו יודעים איך לבנות מכונה שמדפיסה את w - היא פשוט כותבת אותה. למעשה, זה כל כך פשוט לבנות אותה, שאנחנו יכולים לבנות מכונה שבהנתן מילה w, היא יודעת לבנות מכונה שתדפיס את w.
פעולת B: קוראת את מה שכתוב על הסרט, x, וכותבת בעזרת M את $\langle p_x \rangle \cdot x$.
מההגדרה, $A = p_{\langle B \rangle}$.
אחרי ריצת A, כתוב $\langle B \rangle$ על הסרט. כשנריץ את B, יהיה כתוב על הסרט:

$$\left\langle p_{\langle B \rangle} \right\rangle \cdot \langle B \rangle$$

אבל זה בדיוק $\langle A \rangle \cdot \langle B \rangle$, ו"סיימנו", אבל לא באמת:
נגדיר פעולה של B': קוראת את הסרט x, אם x קידוד של מ"ט, B' מדפיסה את הקידוד:

$$\langle p_x \cdot x \rangle$$

חישוב קידוד בינארי מאונרי

נתון סרט שתכתוב עליו מספר באונרית, נרצה לחשב את הקידוד שלו בבינארית.
הפתרונות הנאיביים לוקחים הרבה זמן. נציין דרך יעילה:
"סוחבים" את המונה במקום ללכת הלוך חזור - זה יקח $O(nlogn)$.

תרגול 6

RAM-Machines

עוד תחנה בנסיון השכנוע ש"מכונת טיורינג זה כמו מחשב".

הגדרה 4.45 מכונת RAM מקבלת סדרה של מספרים $a_1, a_2, ..., a_k$, ומורכבת מ:

- סדרת פקודות P.
- מערך זיכרון מאונדקס על $\mathbb{N}$, ובכל תא אפשר לכתוב מספר מ־$\mathbb{N}$.
- תא מיוחד שנקרא PC.

- תא עבודה 0.

וכן נגדיר את $C(i)$ להיות תוכן התא ה־i במערך.

הרעיון - יש לנו מערך אינסופי של זיכרון (לאורך ולרוחב), תא 0, תא PC וסדרת הפקודות P מגיעה מהרשימה הבאה:

- $\text{load}(op)$ - טוען ל־$C(0)$ את התא op.
- $\text{store}(op)$ - שומר בתא op את $C(0)$.
- $\text{add}(op)$ - $C(0) \leftarrow C(0) + op$.
- $\text{sub}(op)$ - $C(0) \leftarrow max\{C(0) - op, 0\}$.
- $\text{read-to}(op)$ - קורא את איבר הקלט הבא לתא op.
- $\text{jump}(op)$ - $PC \leftarrow op$.
- $\text{jump-if}{>}0(op)$ - if $C(0) > 0$ then $PC \leftarrow op$.
- acc,rej.

כאשר op יכול להיות קבוע, כתובת או מצביע למספר.

איך מממשים את זה בעזרת מכונת טיורינג?

תהי $A_{RAM} = \{\langle P, w\rangle \,|P \text{ is a RAM machine that accepts } w\}$.
מה זה אומר ש־P מקבלת את w? אם מתישהו מגיעים ל־acc או rej (ולא נכנסים ללופ אינסופי).

טענה 4.46 קיימת מ"ט U כך ש: $L(U) = A_{RAM}$.

הוכחה: נבנה את U כך:
סרט 1: נכתוב את P (הפקודות).
סרט 2: מילת הקלט w (סדרת מספרים).
סרט 3: PC.
סרט 4: ה־RAM (הזיכרון).
סרט 5: חישובים.
בכל צעד, U תקרא את הפקודה המתאימה ל־PC, ותפעל בהתאם. ■

למשל (בהתחלת ריצה PC מאותחל ל־1), ריצה המתחילה כך:

$$1: read-to(8)\ \#add(2)\ \#jump(7)\ \#$$
$$2: 42\#6\#11\#395$$
$$3: 1$$
$$4: (0,0)$$

לאחר שני צעדים הסרטים יראו כך:

$$1: read-to(8)\ \#add(2)\ \#jump(7)\ \#$$
$$2: 6\#11\#395$$
$$3: 2$$
$$4: \begin{pmatrix} \overbrace{0}^{cell\#}, \overbrace{2}^{value} \end{pmatrix} \# \begin{pmatrix} \overbrace{8}^{cell\#}, \overbrace{42}^{value} \end{pmatrix}$$

מבחינה פרקטית אין כאן הרבה. מבחינה תיאורתית, זה מראה שלא נעשה עם מחשב משהו שאי אפשר לעשות עם מכונת טיורינג.

רדוקציות

תזכורת:
יהיו $L_1, L_2 \subseteq \Sigma^*$, נאמר ש־$L_1$ ניתנת לרדוקציית מיפוי ל־L_2, ונסמן $L_1 \leq_M L_2$, אם קיימת מכונת טיורינג T כך שבהנתן קלט x, T תמיד עוצרת עם $T(x)$ כתוב על הסרט, ומתקיים:

$$x \in L_1 \Leftrightarrow T(x) \in L_2$$

ההגדרה הזו מאפשרת לנו להוכיח (בין היתר) את המשפט הבא:

משפט 4.47 אם $L_1 \leq_M L_2$, ו־$L_2 \in RE$, אזי גם $L_1 \in RE$.

מדוע זה נכון? אם היתה מכונה שמזהה את L_2, נוכל לבנות ממנה מכונה שתזהה את L_1:
המכונה החדשה M תעביר את הקלט x דרך T, ואז דרך המכונה שמזהה את L_2, ו־M תחזיר rej אם המכונה שמזהה את L_2 דחתה, ותחזיר acc אם המכונה שמזהה את L_2 קיבלה.
מה שיותר מעניין הוא שלילת המשפט:

משפט 4.48 אם $L_1 \leq_M L_2$, ו־$L_1 \notin RE$, אז $L_2 \notin RE$.

המשפט הזה שימושי (לתרגיל!) להוכחה ששפה מסויימת לא ב־RE (או R או $co-RE$), אם אנחנו יודעים זאת לגבי השפה השניה. כמובן שצריך להראות שיש מכונה T שמהווה רדוקציית מיפוי (ולא רק להגיד שקיימת כזו - נשים לב להבדל הזה בהמשך), וזה לעיתים לא קל בכלל.

דוגמאות

1.

$$EVEN_{TM} = \{\langle M\rangle \,|\, L(M) = \{x| \,|x| \text{ is even}\}\}$$

השפה $L(M)$ היא למעשה מאוד פשוטה, השפה הכללית יותר מסובכת. יתרה מזו, $L(M)$ היא רגולרית.
תחילה, יש לקבל אינטואיציה, אבל היא לא תמיד נכונה. נראה לנו כשמביטים בזה כמה דקות כי היא לא ב־R, כי "צריך להריץ הרבה דברים, זה נראה מסובך". האם היא ב־$co-RE$? נטען שהיא אינה ב־$co-RE$, כלומר שלא ניתן לעצור בשלב מסויים של בדיקה של מכונה ולהגיד שאינה בשפה. זאת, כי יש מילים זוגיות שלא נעצור עליהן. נוכיח זאת באופן פורמלי:

טענה 4.49 $A_{TM} \leq_M EVEN_{TM}$.

איך נוכיח זאת?
נבנה T, כך שבהנתן $\langle M, w\rangle$, T תפלוט קידוד של מכונה H כך שאם M מקבלת את w, אז $L(H) = \{y| \,|y| \text{ is even}\}$ (כלומר $\langle H\rangle \in EVEN_{TM}$), ואם M לא מקבלת את w, אז $L(H) \neq \{y| \,|y| \text{ is even}\}$.
נבנה את H, הפלט של T, באופן הבא:
תחילה, על המכונה שלנו לבדוק אם האורך של y זוגי. יש אוטומט קבוע שעושה את זה, כך שאין בעיה.
אם $|y|$ זוגי, נריץ את M על w. אם M קיבלה, נקבל, אם M דחתה, נדחה.
אם $|y|$ אי זוגי, נדחה.
אין בעיה להדפיס מכונה כזו, נשים לב כי אנחנו **לא טוענים** שאם נריץ את המכונה הזו היא לא תתקע!

הוכחה: נבנה רדוקציית מיפוי T מ־A_{TM} ל־$EVEN_{TM}$:

בהנתן קלט $\langle M, w\rangle$, T תפלוט את $\langle H\rangle$, כאשר H היא מכונה הפועלת כך:

בהנתן קלט y, H בודקת האם $|y|$ זוגי. אם כן, H מריצה את M על w, ועונה "אותו דבר".

אם לא, H דוחה.

T חשיבה (ניתנת לחישוב) כי קל להדפיס קידוד של מ"ט הבודקת זוגיות, וקל לכתוב את הקידוד של M כך שתרוץ על w.

נותר להוכיח נכונות - אם $\langle M, w\rangle \in A_{TM}$, אזי M מקבלת את w. לכן, עבור קלט y ל־H, אם $|y|$ אז H דוחה, ואם $|y|$ זוגי, אז H תקבל את y כי $M(w)$ יחזיר acc, ולכן $L(H) = \{y| \,|y| \text{ is even}\}$, ולכן $\langle H\rangle \in EVEN_{TM}$.

אם $\langle M, w\rangle \notin A_{TM}$, אז M לא מקבלת את w, ולכן $L(H) = \phi$, שכן אם $|y|$ אי זוגי, אז H דוחה, ואם $|y|$ זוגי, אז H מתנהגת כמו M על w, ולכן $\langle H\rangle \notin EVEN_{TM}$. ■

מסקנה 4.50 על כן $EVEN_{TM} \notin co-RE$.

נסמן:

$$A_{\overline{TM}} = \{\langle M, w\rangle \,|M \text{ does not accept } w\} \notin RE$$

(אנו יודעים את זה כי $A_{TM} \notin R$, אבל $A_{TM} \in RE$, ולכן בהכרח משאלה 2 תרגיל 6, $A_{TM} \notin co-RE$).

טענה 4.51 $A_{\overline{TM}} \leq_M EVEN_{TM}$.

הבעיה כאן היא כזו: אם נקבע את הפלט של הרדוקציה להריץ את M על w, "זה פרוע", אנחנו לא יודעים אם אפשר להתקדם.
הרעיון הוא לרסן את הריצה של M על w עד שלב מסויים. כמה ניתן לה לרוץ? 7000 צעדים לא ממש עוזרים לנו, היא לא אומרת לנו שום דבר על התשובה של w. נרצה לתת לה לרוץ יותר ויותר על w. ה"טריק" פה הוא כזה:

הוכחה: נבנה רדוקציית מיפוי T שבהנתן קלט $\langle M, w\rangle$ תפלוט את הקידוד של מ"ט H הפועלת באופן הבא:

H, בהנתן קלט y, תריץ את M על w למשך $|y|$ צעדים. לאחר מספר הצעדים הללו המכונה יכולה לקבל, או שלא הגיעה ל־acc. זה יהווה את הפילטר שאנחנו צריכים.
אם המכונה M קיבלה את w, נדחה.
אחרת, נבדוק את $|y|$ זוגי. אם לא, נדחה, אם $|y|$ זוגי, נקבל.
"למה זה עובד"?

על קלטים y גדולים מספיק יהיו מספיק צעדים כדי לקבל את w (שכן ריצה מקבלת אורכה סופי), במידה ויש לה ריצה מקבלת. ■

2. נביט בשפה:

$$USELESS = \{\langle M\rangle \,|\text{exists q}\in\text{M s.t. no run reaches it}\}$$

נשים לב כי $USELESS \in co-RE$.

טענה 4.52 $USELESS \notin RE$.

הוכחה: נראה רדוקציה מ־$A_{\overline{TM}}$:

בהנתן קלט y ־ T תחזיר את הקידוד של המכונה H, אותה היא בונה באופן הבא (נשנה אותה מעט בסוף):

H מריצה את M על w. אם M מקבלת את w, H עוברת למצב חדש q_{trev}, ממנו עוברים על כל המצבים, ובסוף נקבל. אם M דוחה את w, H דוחה.

כעת, אם $\langle M, w\rangle \in A_{\overline{TM}}$, אזי M לא מקבלת את w, ואז H לא מגיעה למצב המיוחד, על כן $\langle H\rangle \in USELESS$.
אם $\langle M, w\rangle \notin A_{\overline{TM}}$, M מקבלת את w, H מגיעה למצב המיוחד, ועוברת על כל המצבים, לכן $\langle H\rangle \notin USELESS$.

נותר להסביר כיצד בונים את q_{trev}: הרעיון הוא לכתוב על הסרט סמל מיוחד @, ומכל מצב ב־H להוסיף מעבר למצב "הבא" בעת קריאת @ (לאחר סידור שרירותי של המצבים) ללא הזזת הראש. המצב הזה כותב את הסמל המיוחד @ על הסרט, ומתחיל את המעבר.
הגדרה זו מבטיחה ביקור בכל מצב, וכן כי לא יקרה מצב שכזה לפני הגעה למצב המיוחד.
עולה כאן בעיה נוספת ־ יש לוודא כי קיים קלט עליו H תבקר ב־q_{rej}, אבל זה לא יכול להיות חלק מהמעבר. לצורך זה, משנים את H כך שקיים קלט ספציפי x (למשל, $x = \varepsilon$) כך שבריצת H עליו, היא עוברת ישירות ל־q_{rej}. ■

תרגול 7

מכונת טיורינג אי־דטרמיניסטית (NTM)

9/5/2012

הגדרה 4.53 מכונת טיורינג אי דטרמיניסטית (NTM) זהה למ"ט רגילה (TM), מלבד שינוי בפונקציית המעברים שלה:

$$\delta : Q \times \Gamma \to 2^{Q\times\Gamma\times\{R,L\}}$$

כלומר, ריצה של NTM על מילה w היא סדרת קונפיגורציות (כמו ב־TM) כך שלכל קונפיגורציה עוקבת לזו שלפניה לפי δ.
נאמר ש־N NTM מקבלת מילה w אם קיימת ריצה מקבלת של N על w.
השפה של N NTM היא:

$$L(N) = \{w | N \text{accepts } w\}$$

ונאמר ש־N מזהה את $L(N)$.
נסתכל על:

$$C = \{\langle n \rangle | n \text{not primal}\}$$

NTM שמזהה את C תכתוב באופן אי דטרמיניסטי על הסרט את כל המספרים, ועבור כל מספר (שאינו 1 או n), נבדוק האם הוא מחלק את n.

משפט 4.54 תהי $L \subseteq \Sigma^*$. L ניתנת לזיהוי ע"י NTM אמ"מ קיימת שפה K כריעה ($K \in R$) כך ש:

$$L = \{x | \exists y \text{ s.t. } x\#y \in K\}$$

לדוגמא, עבור C שהגדרנו מעלה, נתבונן בשפה:

$$K = \left\{ \overbrace{\langle n, p \rangle}^{n\#p} \middle| \, p|n \right\}$$

מתקיים ש־$\langle n \rangle \in C$ אמ"מ קיים p כלשהוא כך ש־$p|n$, כלומר כך ש: $n\#p \in K$, ולכן:

$$C = \{\langle n \rangle | \exists p, \, \langle n, p \rangle \in K\}$$

ונשים לב כי K כריעה.
כעת, נוכיח את המשפט: **הוכחה:** $\Leftarrow$ נניח כי L ניתנת לזיהוי ע"י NTM. לכן, קיימת N NTM כך ש־$L = L(N)$. תהי:

$$K = \{x\#y | \, y \text{describes an accepting run of } N \text{on } x\}$$

אזי, נשים לב כי $L = \{x | \exists y \; s.t. \; x\#y \in K\}$, כי אם $x \in L$, קיימת ריצה חוקית מקבלת r של N על x. אזי נסמן $y = \langle r \rangle$. ואז מההגדרה $x\#y \in K$. באופן דומה, אם קיים y כך ש־$x\#y \in K$, אז y מקודדת ריצה מקבלת של N על x, ועל כן $x \in L$.
לסיום, נשאר להראות כי $K \in R$ - פשוט בודקת ש־y היא ריצה חוקית ומקבלת של N על x. חוקית - מוודאת נכונות לפי פונקציית המעברים δ, ומקבלת - מוודאת כי המצב הסופי הוא מצב מקבל. על כן, K כריעה, כנדרש.
$\Rightarrow$ נניח שקיימת $K \in R$, וכן כי:

$$L = \{x | \exists y \text{s.t. } x\#y \in K\}$$

נרצה לבנות מכונה אי דטרמיניסטית N שתזהה את L. נבנה אותה באופן הבא:
N "תנחש" (= תכתוב באופן אי דטרמיניסטי את y - כלומר, תנסה את כל האפשרויות בו זמנית), ותוודא ש־$x\#y \in K$ ע"י מכונה המכריעה את K, וסיימנו. ■

טענה 4.55 תהי N NTM, אזי קיימת D TM כך ש־$L(N) = L(D)$.

הוכחה: מהמשפט הקודם, קיימת שפה כריעה K, כך ש:

$$L(N) = \{x | \exists y \text{s.t. } x\#y \in K\}$$

תהי M מ"ט המכריעה את K. נבנה את D באופן הבא:
בהנתן קלט x, D תדפיס בסדר כלשהוא (enumerates) כל $y \in \Sigma^*$ סופי. לאחר מכן, לכל y שכזה, תבדוק האם $x\#y \in L(M) = K$ ע"י הרצת $x\#y$ על M. אם עבור y כלשהוא M מקבלת, D תקבל, אחרת D לא תעצור.
מכיוון ו־M מכריעה, בכל ריצה על y מסויים M תעצור, ואם לא תקבל תמשיך הלאה ל־y הבא, כל כן D תרוץ על כל y אפשרי. ■

נשים לב כי העלות של התהליך מעלה היה מאוד יקר. לא יודעים היום אם ניתן לבצע את המעבר מ־NTM ל־TM בצורה יעילה.

$Re - Recursively\ Enumerable$

נחזור למודלים דטרמיניסטיים.
אנמורטור הוא "מ"ט עם מדפסת":
קיימים סרט קריאה, סרט כתיבה בלבד, ומצב q_{print}. *ציור*.
ברגע שעוברים ל־q_{print}, מדפיסים את הכתוב על הסרט התחתון ומנקים את הסרט העליון.
קצת יותר פורמלית:

- מ"ט עם שני סרטים, הסרט השני לכתיבה בלבד.
- לא מקבל קלט, דהיינו בתחילת הריצה שני הסרטים ריקים.
- אין מצב מקבל ומצב דוחה, אך יש מצב q_{print} שגורם למכונה להדפיס את התוכן של הסרט השני ולנקות אותו.
- הריצה של אנומטור מדפיסה סדרת מילים.
- השפה של אנומטור היא אוסף המילים שהוא מדפיס:

$$L(E) = \{w | E \text{eventually prints } w\}$$

משפט 4.56 $L \in RE$ אמ"מ קיים אנומרטור E כך ש: $L(E) = L$.

הוכחה: $\Leftarrow$ תהא $L \in RE$, נבנה אנמואטור E כך ש: $L(E) = L$. תהי M מ"ט המזהה את L.
E יריץ במקביל את M על כל המילים. כל פעם שמילה תתקבל, E ידפיס אותה. אזי השפה של E היא $L(E) = L(M)$.
$\Rightarrow$ יהי E אנומרטור כך ש: $L(E) = L$. נבנה מ"ט M כך ש־$L(M) = L$:
בהנתן קלט x, M תסמלץ את ריצת E על הקלט, וכל פעם ש־E מדפיסה מילה, M משווה אותה ל־x. אם x הודפסה, M תקבל. אחרת היא לא תעצור. אזי $L(M) = L(E) = L$. ■

למה אנמורטורים טובים?

טענה 4.57 לכל שפה $L \in RE \backslash R$ קיימת שפה $K \subseteq L$ כך ש־K כריעה ואינסופית.

הוכחה: L אינסופית, שכן אחרת היתה רגולרית, ובפרט ב־R.
מהמשפט הקודם, קיים E אנומרטור כך ש־$L(E) = L$. נגדיר:

$$K = \{w | E \text{ does not print words longer that } w \text{before } w\}$$

נראה כי שפה זו עומדת בדרישות:

- $K \subseteq L$ - מהגדרתה.
- K אינסופית - נניח בשלילה כי K סופית. אזי, קיימת $K \ni w$ בעלת אורך מקסימלי. מההגדרה מילה ארוכה מ־w לא מודפסת לפני w, אבל נשים לב כי לאחר ש־E מדפיס את w, הוא אינו יכול להדפיס מילים ארוכות יותר ממנה - שכן אז המילה הראשונה שכזו ב־K מהגדרתה, בסתירה למקסימליות האורך של w. אבל, אז L סופית, שכן מספר המילים ש־E מדפיס סופי, בסתירה להנחה.
- K כריעה: נבנה מ"ט M בצורה הבאה:
בהנתן קלט w, M מסמלצת את ריצת E, ואז משווה את המילים המודפסות ל־w. אם w הודפסה, M תקבל. אם הודפסה מילה ארוכה מ־w, M תדחה. M מכריעה את K, שכן לכל קלט w, או ש־E מדפיסה אותה, או שהיא מדפיסה מילה ארוכה ממנה (לפני שהדפיסה את w) בשלב מסויים (מכיוון ו־L אינסופית), ועל כן M תמיד מקבלת או דוחה. שנית, אם M מקבלת את w אמ"מ E מדפיסה רק מילים שאורכן לא גדול מאורך w לפני w. על כן, $L(M) = K$.

■

תרגול 8

23/05/2012

בהרצאה ראינו:
$TIME(f(n))$ הוא אוסף השפות L, עבורן קיימת מ"ט המכריעה את L ורצה בזמן $O(f(n))$.
כמו כן, הגדרנו:

$$P = \bigcup_{k>1} TIME(n^k)$$

רדוקציות

ראינו רדוקציית מיפוי, ואיך היא עוזרת להוכיח אם שפה מסויימת היא ב־R, RE או $co-RE$.
באופן דומה נגדיר עבור ריצות בזמן פולינומי:

הגדרה 4.58 רדוקציית מיפוי פולינומית משפה L_1 לשפה L_2 היא מ"ט T כך ש:

1. לכל קלט x, T עוצרת בריצתה על x כאשר כתוב $T(x)$ על הסרט.
2. מתקיים:

$$x \in L_1 \Leftrightarrow T(x) \in L_2$$

3. T רצה בזמן פולינומי.

נסמן $L_1 \leq_p L_2$.

נרצה להוכיח משפט הדומה למה שראינו עבור רדוקציית מיפוי כללית:

משפט 4.59 אם $L_1 \leq_m L_2$, ו־$L_2 \in P$, אז $L_1 \in P$.

הוכחה: תהי M_2 מ"ט המכריעה את L_2 בזמן f_2 פולינומי, ותהי T רדוקציה פול' הרצה בזמן f_T.
זמן הריצה של המכונה המכריעה את L_1 הוא $O(f_2(f_T(n)) + f_T(n))$, והוא פולינומי. ■

עכשיו צריך ללמוד איך משתמשים ברדוקציה הזו.
נתחיל לאט ונתקדם, עד סוף השיעור כבר נגיע לרדוקציה מסובכת.
עם זאת, העולם פה מאוד שונה ממה שהתעסקנו בו עד כה בענייני רדוקציות, שכן כמעט כל מה שראינו לא היה כריע, ושאול מבטיח שהנושא יהיה גם מאוד יפה.
בעת הכתיבה, נחלק לשלושה חלקים:

1. תיאור הרדוקציה T.
2. הוכחה מדוע המכונה T היא אכן רדוקציה.
3. ניתוח זמן הריצה.

טענה 4.60 $A_{TM} \leq_p HALT_{TM}$.

הוכחה: :

1. **בנייה:** בהנתן $\langle M, w \rangle$, הרדוקציה T תפלוט את $\langle H, w' \rangle$ כך ש־H מתקבלת מ־M ע"י החלפת q_{rej} בלולאה, ו־$w' = w$.
2. **נכונות:** כבר ראינו, אז נדלג.
3. **ניתוח זמן הריצה:** פרסור המידע - אף פעם לא חישבנו באמת, ,אבל זה לא יותר מ־$\left|\langle M^3 \rangle\right|$ (בדיקה שכן המצבים והאותיות חוקיות). כדי להוסיף לולאה צריך לעבור על טבלת המעברים ולהוסיף לה שורה. צעד זה לוקח לכל היותר $O\left(\left|\langle M \rangle\right|^2\right)$. כמו כן, העתקת w לוקחת $O\left(\left\langle |w|^2 \right\rangle\right)$. לכן בסה"כ הרדוקציה פולינומית.

■

הגדרה 4.61 יהי $G = (V, E)$ גרף לא מכוון.

1. קליקה ב־G היא קבוצה $S \subseteq V$ כך שלכל $u, v \in S$ מתקיים $\{u, v\} \in E$ (תת גרף מלא).
2. Vertex Cover ב־G הוא קב' $L \subseteq V$ כך שלכל $e \in E$ קיים $v \in L$ כך ש־$v \in e$.

נכיר שתי שפות מפורסמות:

הגדרה 4.62

$$CLIQUE = \{\langle G,k\rangle | \text{there exist a k-clique in } G\}$$

הגדרה 4.63

$$VC = \{\langle G,k\rangle | \text{there exist a Vertex Cover of size k in } G\}$$

טענה 4.64 $CLIQUE \leq_p VC$.

הוכחה: שוב, לפי השלבים:

1. **בנייה:** הרדוקציה T תפעל כך: בהנתן $\langle G,k\rangle$, $G=(V,E)$, T תפלוט את הזוג $\langle \overline{G}, |V|-k\rangle$, כאשר $\overline{G} = \langle V, \overline{E}\rangle$.
2. הפעם נוכיח קודם **זמן ריצה:** הרדוקציה תעבור על טבלת הצלעות ותהפוך את כל התאים בה - כלומר $O\left(|V|^2\right)$ "הפיכות", כל אחת לוקחת זמן פולינומי. כמו כן, חישוב $|V|-k$ יקח $O(|V|\,log\,|V|)$. סה"כ זמן פול'.
3. **נכונות:** $\Leftarrow$ נניח ש-$\langle G,k\rangle \in CLIQUE$. תהי $C \subseteq V$ k-קליקה ב-G. תהי $D = V\backslash C$. נטען ש-D הוא VC ב-$\overline{G}$: תהי $\{u,v\} \in \overline{E}$. נניח בשלילה ש-$u,v \notin D$. אזי $u,v \in C$. מכיוון ש-$\{u,v\} \notin E$, זו סתירה לכך ש-C קליקה. כמו כן $|D| = |V| - |C| = |V| - k$, ולכן $\langle \overline{G}, |V|-k\rangle \in VC$.
$\Rightarrow$ נניח ש-$\langle \overline{G}, |V|-k\rangle \in VC$. אז תהי $D \subseteq V$ כך ש-$|D| = |V|-k$, ו-D היא VC. תהי $C = V\backslash D$, ונטען ש-C היא k-קליקה ב-G:
יהיו $u,v \in C$. נניח בשלילה ש-$\{u,v\} \notin E$, אז $\{u,v\} \in \overline{E}$, אבל $u,v \notin D$, בסתירה לכך ש-D היא VC. לכן C קליקה, וכן $|C| = |V| - |D| = k$.

■

הגדרה 4.65 נוסחא ב-propositional logic φ היא נוסחא מהצורה:

- פסוקית אטומית x.
- $\psi_1 \vee \psi_2$, כאשר ψ_1, ψ_2 הן נוסחאות.
- $\neg\psi_1$, כאשר ψ_1 היא נוסחא.

דוגמא:

$$\neg\left(x \vee \neg\left(y \vee z\right)\right)$$

הגדרה 4.66 נוסחא φ נקראת ספיקה אם קיימת לה השמה מספקת לפסוקיות האטומיות (כלומר התוצאה המתקבלת מההשמה היא 1).

למשל, הדוגמא מעלה היא ספיקה, למשל ההשמה $x=0, y=1, z=0$ מספק.
עם זאת, הנוסחא: $x \wedge \neg x$ אינה ספיקה.

הגדרה 4.67

$$SAT = \{\langle \varphi\rangle | \, \varphi \text{ is satisfiable}\}$$

הרדוקציה הבאה מסובכת מעט ממה שידרש מאיתנו בתרגיל, אז "אם אתם לא מבינים עד הסוף, זה בסדר".

טענה 4.68 $CLIQUE \leq_p SAT$.

הוכחה: בנייה: בהנתן $\langle G,k\rangle$ הרדוקציה T תפלוט $\langle\varphi\rangle$ הנראית כך:
לכל $1\leq i\leq k$ ולכל $v\in V$ יהיה פסוקית אטומית $y_{v,i}$.
נגדיר:

$$\varphi=\psi_1\wedge\psi_2\wedge\psi_3\wedge\psi_4$$

כאשר:

$$\psi_1=\bigwedge_{1\leq i\leq k}\bigvee_{v\in V}y_{v,i}$$
$$\psi_2=\bigwedge_{1\leq i\leq k}\bigwedge_{u\neq v\in V}\neg\left(y_{u,i}\wedge y_{v,i}\right)$$
$$\psi_3=\bigwedge_{1\leq i<j\leq k}\bigwedge_{u\in V}\neg\left(y_{u,i}\wedge y_{u,j}\right)$$
$$\psi_4=\bigwedge_{1\leq i<j\leq k}\bigwedge_{\{u,v\}\notin E}\neg\left(y_{u,i}\wedge y_{v,j}\right)$$

אינטואיציה:
$y_{v,i}=1$ אומר ש־v הוא הקודקוד ה־i בקליקה ב־G.
ψ_1 - לכל $1\leq i\leq k$, קיים קודקוד v שהוא ה־i בקליקה.
ψ_2 - אין שני קודקודים שונים ששניהם ה־i בקליקה.
ψ_3 - כל קודקוד יכול להיות רק באינדקס אחד בקליקה.
ψ_4 - אם אין צלע $\{u,v\}$, אז u ו־v לא יכולים להיות שניהם בקליקה.

זמן ריצה: יש $|V|-k$ אטומים, והנוסחא בגודל $O\left(|V|^2k^2\right)$, ולכן כתיבתה לוקחת זמן ריצה פולינומיאלי.

נכונות: $\Leftarrow$ נניח שב־G יש k־קליקה. נסמנה $C=\{v_1,v_2,...,v_k\}$. נבנה השמה מספקת ל־φ כך:
$y_{v_i,i}=1$ לכל $1\leq i\leq k$, והשאר 0.
ψ_1 מסופקת, כי לכל $1\leq i\leq k$, $y_{v_i,i}=1$.
ψ_2 מסופקת, כי לכל $1\leq i\leq k$, קיים ויחיד $y_{v_i,i}$ שערכו 1.
ψ_3 מסופקת, כי לכל קודקוד v, יש לכל היותר אינדקס אחד j כך ש־$y_{v,j}=1$.
ψ_4 מסופקת, כי לכל זוג $v_i,v_j\in C$, קיימת הצלע $\{v_i,v_j\}\in E$, ולכן קודקודים אלו לא יופיעו בנוסחא.
$\Rightarrow$ נניח ש־φ ספיקה, ותהי π השמה מספקת.
מ־ψ_1,ψ_2,ψ_3 נובע שבדיוק k משתנים מושמים ל־1. נסמנם בו־$y_{v_1,1},...,y_{v_k,k}$. לפי ψ_2,ψ_3, כל ה־v_i־אים שונים. לפי ψ_4, לא קיימים $\{v_i,v_j\}\notin E$, ולכן $\{v_1,...,v_k\}$ היא k־קליקה. ■

תרגול 9

תרגול 10

מכונת טיורינג אוניברסלית

נזכור כי קיימת מ"ט U כך שבהנתן קלט $\langle M,w\rangle$, U "מסמלצת" את ריצת M על w.
אינטואיציה: גיימבוי הוא מכונת טיורינג אוניברסלית!
נזכר כיצד היא עובדת, עם טייפ אחד:
על הטייפ בתחילת הריצה רשום הקידוד של M, $\#$ ו־w, שהוא בא"ב של U. כיצד מתבצעת הסימולציה?
U כותבת משמאל $\#$ ואז את המצב הנוכחי. U רושמת "מעל" האות הראשונה אות מסויים, למשל נקודה (מוסיפה לה סימן), משווה את האות הנוכחית עם הקידוד של δ עד שמוצאת את המצב, כותבת את האות החדשה במקום w_1, מוסיפה נקודה מעל w_2, וחוזר חלילה.
נשאל, מהי סיבוכיות המקום של הריצה הזו? כלומר, בכמה מקום U משתמשת כדי למסלץ את M על w?
נניח ש־M מתשתמשת ב־n תאים בריצתה על w. אז U משתמשת בכל ב־$|M|+n\cdot|\langle M\rangle|=O\left(n\cdot|\langle M\rangle|\right)$, שכן הקידוד של w הוא בא"ב של U, ולכן "בקידוד הכי מטומטם שאפשר לחשוב עליו", מכיוון והקידוד של הא"ב עבודה של M נמצא ב־$\langle M\rangle$, נקבל את סיבוכיות המקום הזו.
נשים לב לתוצאה המעניינת שקיבלנו - סיבוכיות המקום לינארית. נשתמש בזה בתרגול הבא.

סיבוכיות מקום

נרצה להגדיר סיבוכיות מקום.
נסיון ראשון+שני:

הגדרה 4.69 נאמר שמ"ט M רצה במקום $f(n)$ אם לכל w כך ש־$|w|=n$, M משתמשת בלכל היותר $f(n)$ תאים בריצתה על w, ותמיד עוצרת.

כאשר כבר ראינו שאם מספר התאים מוגבל, אזי כך גם מספר המצבים, לכן קל לבדוק אם המכונה נתקעת, ו"מאוד הגיוני" לדרוש שהמכונה אכן תעצור (אחרת המכונה תכתוב אינסוף פעמים על אותם התאים, וזה לא "מסתדר" עם מה שאנחנו רוצים להגדיר ־ סיבוכיות מקום).
נביט בשפה:

$$L=\left\{ w\in\{a,b\}^{*}\middle|\, w\text{ends with }a\right\}$$

נרצה שהזיכרון שנשתמש בו קטן מהקלט. כיצד נביט זאת?
נעבוד עם מ"ט עם שני סרטים ־ סרט 1 מכיל את הקלט, והוא לקריאה בלבד. סרט 2 ־ עבודה.

הגדרה 4.70 נאמר שמ"ט M עובדת ב־$f(n)$ זכרון אם לכל $|w|=n$, M עוצרת בריצתה על w תוך שימוש בלכל היותר $f(n)$ תאים בסרט 2.

השלב הבא ־ הגדרה במכונה אי דטרמיניסטית:

הגדרה 4.71 נאמר שמ"ט א"ד N רצה במקום $f(n)$ אם לכל $|w|=n$, כל הריצות של N על w עוצרות תוך כדי שימוש בלכל היותר $f(n)$ תאים בסרט 2.

הגדרה 4.72

$$SPACE(f(n))=\{L|L\text{is decidable by a TM that runs in }O(f(n))\text{space}\}$$
$$NSPACE(f(n))=\{L|L\text{is decidable by a non-deterministic TM that runs in }O(f(n))\text{space}\}$$

ועכשיו, נעבור למחלקות ה"מעניינות":
עם זמנים קצרים לא היה לנו הרבה מה להתעסק (יש משפט שאומר שאם מ"ט רצה בזמן קצר אזי השפה שלה רגולרית). אולם, יש סיבה לדון בסיבוכיות מקום שכזו.

הגדרה 4.73

$$PSPACE=\bigcup_{k>0}SPACE\left(n^{k}\right)$$
$$NSPACE=\bigcup_{k>0}NSPACE\left(n^{k}\right)$$

דוגמאות דוגמא מפתיעה:

$$CNF-SAT\in PSPACE$$

הוכחה: בהנתן נוסחא φ, מ"ט דט' תכתוב את כל ההשמות האפשריות ל־φ, אחרת אחר השניה, ועבור כל השמה תבדוקה האם היא מספקת, בזמן פולינומי, ולכן גם במקום פולינומי. כתיבת השמה אחת לוקחת $|\varphi|$ תאים, ולכן סה"כ מקום פולינומי. ■

זוהי דרך אופיינית ל־$PSPACE$ ־ כתיבת כל האפשרויות ובדיקתן.

הערה 4.74 אבחנות מעניינות:

1. $P\subseteq PSPACE$, כי בזמן פולינומי ניתן לכל היותר להשתמש במספר פולינומי של תאים.

2. $NP \subseteq NPSPACE$, כנ"ל.

3. $NP \subseteq PSPACE$.

הוכחה: תהי $L \in NP$, אזי $L \leq_p SAT$. תהי T רדוקציה פולינומית כנ"ל. אזי אלגוריתם שמכריע את L ב־$PSPACE$ יפעל כך:

בהנתן x, חשב את $T(x)$, והרץ אלגוריתם $PSPACE$ ל־SAT על $T(x)$.

מכיוון ש־T פולינומית, גם האורך של $T(x)$ פולינומי ב־$|x|$, ולכן כל אלגוריתם רץ במקום פולינומי ב־$|x|$. ■

טענה 4.75 $SPACE \subseteq TIME\left(n \cdot 2^{O(f(n))}\right)$.

הוכחה: מ"ט שתמיד עוצרת לא חוזרת על קונפיגורציה, לכן זמן הריצה של מכונה הרצה במקום $f(n)$ חסום ע"י מספר הקונפיגורציות שלה, שהוא לכל היותר:

$$\underbrace{|Q|}_{state} \cdot \underbrace{|\Gamma|^{f(n)}}_{tape2} \cdot \underbrace{f(n)}_{\text{location of the 2nd head}} \cdot \underbrace{|n|}_{\text{location of the 1st head}} \overset{algebra}{=} n \cdot 2^{O(f(n))}$$

כאשר $|\Gamma|$ אפשרויות לכל תא, ו־$f(n)$ תאים, הסרט הראשון לקריאה בלבד. ■

4. $PSPACE \subseteq EXP$, $NPSPACE \subseteq NEXP$.

[44]

השאלה הטבעית:

$$PSPACE \overset{?}{=} NPSPACE$$

אנחנו מקווים שאפשר להוכיח משהו בנוגע לזה. ננסה את הפתרון הנאיבי:

נסיון: נעשה דטרמיניזציה.

בהנתן N הרצה במקום א"ד $f(n)$, נבנה M דטרמיניסטית:

נביט בעץ הריצות, ונמספר אותן. ריצה מקודדת ע"י סדרה של מספרים, ואורך הסדרה כאורך ריצה. אורך ריצה מקסימלי הוא $2^{O(f(n))}$ - אקספוננציאלי ב־$f(n)$, לכן הגישה הנאיבית נכשלת.

משפט 4.76

$$NPSPACE = PSPACE$$

זוהי אחת מהתוצאות הראשונות בחישוביות שהיא "באמת חשובה".

אנחנו נוכיח משפט יותר חזק:

משפט 4.77 עבור $f(n) = \Omega(logn)$, $NSPACE \subset E(f(n)) \subseteq SPACE\left(f^2(n)\right)$.

הוכחה: נוכיח תחת ההנחות:

1. $f(n) \geq n$.

2. f חשיבה המקום, כלומר, קיימת מ"ט M כך שבהנתן 1^n, M פולטת את $f(n)$ בבינארית תוך שימוש ב־$O(f(n))$ מקום[45].

[44] "עזוב אותך מפרקטיקה, 40 שנות עבודה זה $O(1)$" - שאול על דברים שאנחנו לא נדון בהם בקורס.

[45] רב הפונקציות "הפשוטות" שאנו מכירים הן חשיבות במקום.

אינטואיציה:
בהנתן w, נסתכל בגרף הקונפיגורציות של N על w - נסמן ב־c_i. נניח כי קיים c_{acc} אחד (נניח שהמכונה "מנקה" את הסרט לפני מעבר למצב מקבל, ואז יש מצב מקבל יחיד).
N מקבלת את w אמ"מ יש מסלול בגרף מ־c_0 ל־c_{acc}.
יש בגרף $2^{O(f(n))}$ קודקודים (כאן זה המקום היחיד בו הנחה 1 מקלה אלינו, לא נדון מעבר לכך).
אנחנו לא צריכים את המסלול עצמו, אלא רק אם אפשר בכלל להגיע ל־c_{acc}. אם נדע כי אפשר להגיע לקודקוד מסוים מ־c_0, ואם ממנו אפשר להגיע ל־c_{app}, אז אנחנו צריכים לשמור (הרבה) פחות מידע. אפשר לעשות תהליך זה רקורסיבית לכל קודקוד, וזה יספיק!

תהי NTM N הרצה במקום $f(n)$. נתאר את הפרוצדורה $CanYield$ (האם אפשר להגיע ל־c_2 מ־c_1 תוך t צעדים:

$$\begin{aligned}&\text{CanYield}(c_1, c_2, t):\\&\text{if t=1 accept if } c_1 = c_2 \text{ or } c_2 \text{ if reachable from } c_1 \text{by } \delta\\&\text{if t>1 for every conf. } c_m \text{accepts if CanYield}\left(c_1, c_m, \frac{t}{2}\right) \wedge \text{CanYield}\left(c_m, c_2, \frac{t}{2}\right)\\&\text{reject}\end{aligned}$$

בהנתן N ו־w, המכונה M תפעל כך:
ראשית, נניח של־N יש קונפיגורציה מקבלת יחידה על w, אחרת נשנה את N בהתאם. כעת נחשב את $f(n)$ וחסם על מספר הקונפיגורציות של N על w: $2^{d \cdot f(n)}$ עבור d כלשהוא. נסמן $t = 2^{d \cdot f(n)}$.
כעת M תריץ את $\text{CanYield}(c_0, c_{acc}, t)$.
נכונות: N מקבלת את w אמ"מ קיימת ריצה מקבלת באורך לכל היותר t אמ"מ M מקבלת את w.
מקום: נחלק לשלבים:

- חישוב $f(n)$ לוקח $O(f(n))$ מקום, כי f חשיבה במקום.
- חישוב $2^{d \cdot f(n)}$ הוא פשוט.
- ב־$CanYield$:
  - עומק הרקורסיה הוא $log\left(2^{O(f(n))}\right) = O(f(n))$.
  - לכל רמה ברקורסיה (בונים את מחסנית הרקורסיה), צריך לכתוב c_1, c_2, t, c_m. כל קונפיגורציה נשמרת ב־$f(n)$, ו־$O(f(n))$ ל־t, ולכן בסה"כ $O(f(n))$ תאים.

לכן בסה"כ M רצה במקום $O\left(f^2(n)\right)$. ■

נותרנו עם ציור הבא:
איור
נשים לב כי

$$\underbrace{CO - PSAPCE}_{=CO-NPSPACE} = PSPACE = NPSPACE$$

קיבלנו מחלקה לא דטרמיניסטית שסגורה לשלילה.

תרגול 11

משפט ההיררכיה במקום

13/6/2012

מאוד דומה למשפט ההיררכיה בזמן - אבל יותר קצר, וכמו הרבה דברים של מקום, יותר יפה.

משפט 4.78 תהי $S : \mathbb{N} \to \mathbb{N}$ חשיבה במקום, ו־$S(n) \geq \log n$. אזי, קיימת שפה L כך ש־$L \in SPACE[S(n)]$, אבל $L \notin SPACE[o(S(n))]$[46].

[46] כאן אין חלקי כמו במשפט היררכיית הזמן, שכן כאן אין צורך במקום נוסף עבור המכונה המסמלצת.

הוכחה: נעבוד עם מכונת טיורינג אוניברסלית U המסמלצת את ריצת M על w ב־$O\left(k\cdot\left|\langle M\rangle\right|\right)$ תאים, כאשר k הוא מספר התאים ש־M מתשמשת בהם בריצתה על w.

כמו כן, סימולציה של צעד יחיד לוקחת לכל היותר $\left|\langle M\rangle\right|$ תאים.

נגדיר:

$$S':=\frac{S\left(n\right)}{\left|\langle M\rangle\right|},\ n=\left|\langle M,w\rangle\right|$$
$$L=\left\{\left.\langle M,w\rangle\right|M\text{does not accept }\langle M,w\rangle\text{while using no more than }S'\text{cells}\right\}$$

נטען ש־$L\in SPACE\left[S\left(n\right)\right]$ - נבנה מ"ט D הפועלת כדלהלן:

1. D תחשב את $S\left(n\right)$, ותסמן על הסרט $S\left(n\right)$ תאים. אם בשלב כלשהוא, ריצת D חורגת מהסמן, D תקבל.
2. D תחשב חסם על מספר הקונפיגורציות של M על $\langle M,w\rangle$, תכתוב אותו על הסרט, ותשתמש בו כמונה צעדים. גודל החסם הוא $2^{O\left(S'\right)}=2^{O\left(S\left(n\right)\right)}$, ולכן כתיבה בינארית תיקח $O\left(S\left(n\right)\right)$ תאים.
3. D מסמלצת את ריצת M על $\langle M,w\rangle$, כל עוד המונה חוקי.
   (א) אם המונה מתאפס, D תקבל.
   (ב) אם M מקבלת, D תדחה.
   (ג) אם M דוחה, D תקבל.

D מכריעה את L, כי M לא מקבלת את $\langle M,w\rangle$ תוך שימוש ב־S' תאים, אמ"מ M דוחה את $\langle M,w\rangle$ תוך S' תאים, או ש־M רצה יותר מ־$2^{O\left(S\left(n\right)\right)}$ צעדים, או ש־M חורגת בריצתה מ־S' תאים.

D פועלת במקום $O\left(S\left(n\right)\right)$, כי:

1. $S\left(n\right)$ חשיבה במקום, ולכן חישוב $S\left(n\right)$ ניתן לביצוע ב־$O\left(S\left(n\right)\right)$ תאים.
2. כתיבת $2^{O\left(S\left(n\right)\right)}$ בבינארית לוקחת $O\left(S\left(n\right)\right)$ תאים.
3. סימולציה של M ב־S' תאים לוקחת $S'\cdot\left|\langle M\rangle\right|=\frac{S\left(n\right)}{\left|\langle M\rangle\right|}\cdot\left|\langle M\rangle\right|=S\left(n\right)$ תאים.

נניח בשלילה שמ"ט T מכריעה את L ב־$r\left(n\right)=o\left(S\left(n\right)\right)$. אזי קיים n_0 כך שלכל $n>n_0$ מתקיים $r\left(n\right)<\frac{S\left(n\right)}{\left|\langle T\rangle\right|}$[47]. תהי $\left|w\right|=n$, אזי:

- אם T מקבלת את $\langle T,w\rangle$, היא עושה זאת תוך $r\left(\left|\langle T\rangle\right|+n\right)<\frac{S\left(\left|\langle T\rangle\right|+n\right)}{\left|\langle T\rangle\right|}$, ולכן $\langle T,w\rangle\notin L$.
- אם T לא מקבלת את $\langle T,w\rangle$, אז T דוחה את $\langle T,w\rangle$ תוך $r\left(\left|\langle T\rangle\right|+n\right)<\frac{S\left(\left|\langle T\rangle\right|+n\right)}{\left|\langle T\rangle\right|}$ תאים, ולכן $\langle T,w\rangle\in L$.

סה"כ נקבל כי $L\neq L\left(T\right)$, בסתירה. ■

L ו־NL

כשדיברנו על מחלקות $SPACE$, אמרנו כי אם $S\left(n\right)\leq n$, נעבוד עם מכונת טיורינג עם שני סרטים: סרט 1 - קריאה בלבד, המתחיל עם הקלט ו־_ משני צדדיו, אי אפשר לעבור את _, סרט 2 - עבודה.

הגדרה 4.79

$$L=SPACE\left(\log n\right)\ \left(\text{since }\log n^k=k\cdot\log n,\text{ no need for more definitions}\right)$$
$$NL=NSPACE\left(\log n\right)$$

[47] נשים לב כי $\left|\langle T\rangle\right|$ קבוע, שכן אנו מניחים קיומה של מכונה T ספציפית.

ראינו ש:

$$SPACE\left(f\left(n\right)\right)\subseteq TIME\left(2^{O(f(n))}\right)$$
$$NSPACE\left(f\left(n\right)\right)\subseteq NTIME\left(2^{O(f(n))}\right)$$

מכאן:

$$L\subseteq TIME\left(2^{O(\log n)}\right)$$
$$2^{O(\log n)}\text{"}=\text{"}2^{k\cdot\log n}=n^{k}$$
$$\Rightarrow L\subseteq P$$
$$NL\subseteq NP$$
$$L\subseteq NL\nsubseteq NPSPACE=PSPACE$$

ראינו בשבוע שעבר:

$$NSPACE\left(f\left(n\right)\right)\subseteq SPACE\left(f^{2}\left(n\right)\right)$$

ועל כן:

$$NSPACE\subset E\left(\log n\right)\subseteq SPACE\left(\log^{2}n\right)$$

לא ידוע האם $L=NL$.

הגדרה 4.80 משרן $(transducer)$ במקום לוגריתמי הוא מ"ט עם 3 סרטים:

1. סרט 1 - קלט, לקריאה בלבד.
2. סרט 2 - עבודה, מקום $O\left(\log n\right)$.
3. סרט 3 - פלט, לכתיבה בלבד.

הגדרה 4.81 רדוקציית $\leq_L$ היא רדוקציה הניתנת למימוש ע"י משרן לוגריתמי.

הגדרה 4.82 נאמר ששפה A היא $NL-hard$, אם לכל $B\in NL$ מתקיים $B\leq_L A$.

הגדרה 4.83 נאמר ששפה L היא $NL-complete$, אם היא $NL-hard$ וגם $A\in NL$.

משפט 4.84 אם $B\leq_L A$ ו-$A\in L$, אז $B\in L$.

נביא דוגמא לשפה $NL-complete$:

הגדרה 4.85

$$PATH=\left\{\left\langle G,s,t\right\rangle |\,G\text{is a graph and there's a path from }s\text{to }t\right\}$$

טענה 4.86

$$PATH\in NL-complete$$

הוכחה: תחילה, נראה כי $PATH \in NL$:
בהנתן $\langle G, s, t \rangle$, נספור את הקודקודים, ונכתוב את הקידוד של המספר על סרט העבודה. כמו כן, נכתוב את הקידוד של s.
בכל צעד, ננחש באופן אי דטרמיניסטי קודקוד v, ונבדוק האם יש צלע מהקודקוד הנוכחי ל־v. אם כן, נקטין את המונה ב־1 ונעבור ל־v. אם הגענו ל־t, נקבל, אם המונה מתאפס, נדחה (אם אין צלע, נדחה).
על הסרט כתוב בכל רגע $\langle n \rangle$ ושני קודקודים, סה"כ $3 \cdot \log n$.
כעת, נראה כי $PATH \in NL - hard$:
תהי $A \in NL$, נראה $A \leq_L PATH$. תהי M מכונה המזהה את A ב־NL, ונניח של־M יש קונפיגורציה מקבלת יחידה[48]. רדוקציה T מ־A ל־$PATH$ תפעל כך:
בהנתן w, T תפלוט את גרף הקונפיגורציה של M על w כך ־ T תעבור על כל המחרוזות באורך $k \cdot \log |w|$, כאשר $k \cdot \log |w|$ הוא חסם מלמעלה כל ייצוג של קונפיגורציה של M על w.
כל מחרוזת המייצגת קונפיגורציה תפלוט לסרט הפלט כקודקוד. לאחר מכן, T תפלוט את הצלעות ע"י מעבר על זוגות של קונפיגורציות, וכל זוג שהוא צלע עפ"י פונקציית המעברים של M יפלט כצלע. לבסוף, T תפלוט $s = c_0,\ t = c_{acc}$. ■

נשים לב שקל לפתור את $PATH$ ב־P (למשל ע"י BFS), לכן $NL \subseteq P$.

תרגול 12

$L,\ NL$

ראינו בתרגול הקודם רדוקציות מקום לוגריתמי.
הגדרנו רדוקציות כדי להוכיח משפט:

משפט 4.87 אם $A \leq_L B$, ו־$B \in L$, אז $A \in L$.

אם המשפט הזה לא היה נכון, הגדרנו רדוקציות לא טוב. למזלנו, המשפט נכון.
נזכיר משפט אחר:

משפט 4.88 אם $A \leq_p B$ ו־$B \in P$, אז $A \in P$.

על מנת להוכיח אותו, השתמשנו בהרכבת מכונות ־ הרדוקציה T, ו־M_B המכריעה את B בזמן פולינומי. אבל התיאור מטעה ־ צריך לבנות מכונה המכריעה בזמן פולינומי לפי ההגדרות. אזי שתי המכונות המורכבות למעשה "מוכלות" במכונה אחת שמבצעת את ההכרעה.
למה זה לא עובד במקרה של המשפט הנוכחי לגבי רדוקציות מקום לוגריתמי?
משתמשים ביותר מקום בפלט של הרדוקציה (אקספוננציאלי בלוג)! אם כן, יש צורך בטריק חדש ־ נפעיל את T עבור אות, M_B תעבד עבור האות, ואח"כ תבקש מ־T את האות הבאה, וכך הלאה. נפרמל זאת: **הוכחה:** נבנה מכונה K המכריעה את A ב־L ־ בהנתן קלט x, K תפעל כך:

- תריץ את M_B[49] בכל פעם ש־M_B מבקשת את האות ה־i ב־$T(x)$, K תסמלץ את T עד שתפלוט את האות ה־i, ואז תעביר אותה ל־M_B.
- המקום הדרוש:
  - סרט העבודה של M_B ־ $O(\log(T(x))) = O(\log(|x|))$.
  - מיקום הראש הקורא: $O(\log(T(x))) = O(\log(|x|))$.
  - סימולציית T: $O(log(|x|))$.

■

מסקנה 4.89 אם A היא NL־ שלמה, ו־ $A \in L$, אז $L = NL$.

[48] ראינו שאם יש כמה, פשוט לאחר כל קונפי' מקבלת ננקה את הסרט ונעשה את המצב הזה מצב מקבל. כאן אולי גם נצטרך "לאפס" סרט נוסף, אבל כל התהליך הזה לוקח מספר קצוב וקטן של צעדים.
[49] "על $T(x)$", אבל בפועל אות כל פעם.

הערה 4.90 באותו טריק ניתן להוכיח ש:

$$A \leq_L B,\ B \leq_L C \Rightarrow A \leq_L C$$

זוהי לא מסקנה טריוויאלית, בדיוק כמו שהמשפט הנ"ל לא היה טריוויאלי.

רשמית, זהו כל החומר למבחן ־ הידד! עוד חומרים יפים ־ בתרגולים באתר.

חלק III

תשובות לשאלות מבחינות משנים קודמות

מתוך מועד א' שנה שעברה:
תהי $L \in RE \backslash co-RE$, ותהי M כך ש־$L(M) = L$. הוכח\הפרך:

1. לכל n קיימת w, $|w| > n$, כך ש־M מקבלת את w.
2. לכל n קיימת w, $|w| > n$, כך ש־M לא עוצרת על w.

1נכון, כי אחרת השפה סופית, ואם היא סופית היא ב־R ולכן ב־$co-RE$.
2 - השאלה בעצם היא, האם חייב להיות מספר אינסופי של מילים עליהן M לא עוצרת? נכון - אם בשלילה היא לא עוצרת רק על 10 מילים, אפשר לבדוק אם זו אחת מהמילים, ואז M כריעה, בשלילה.