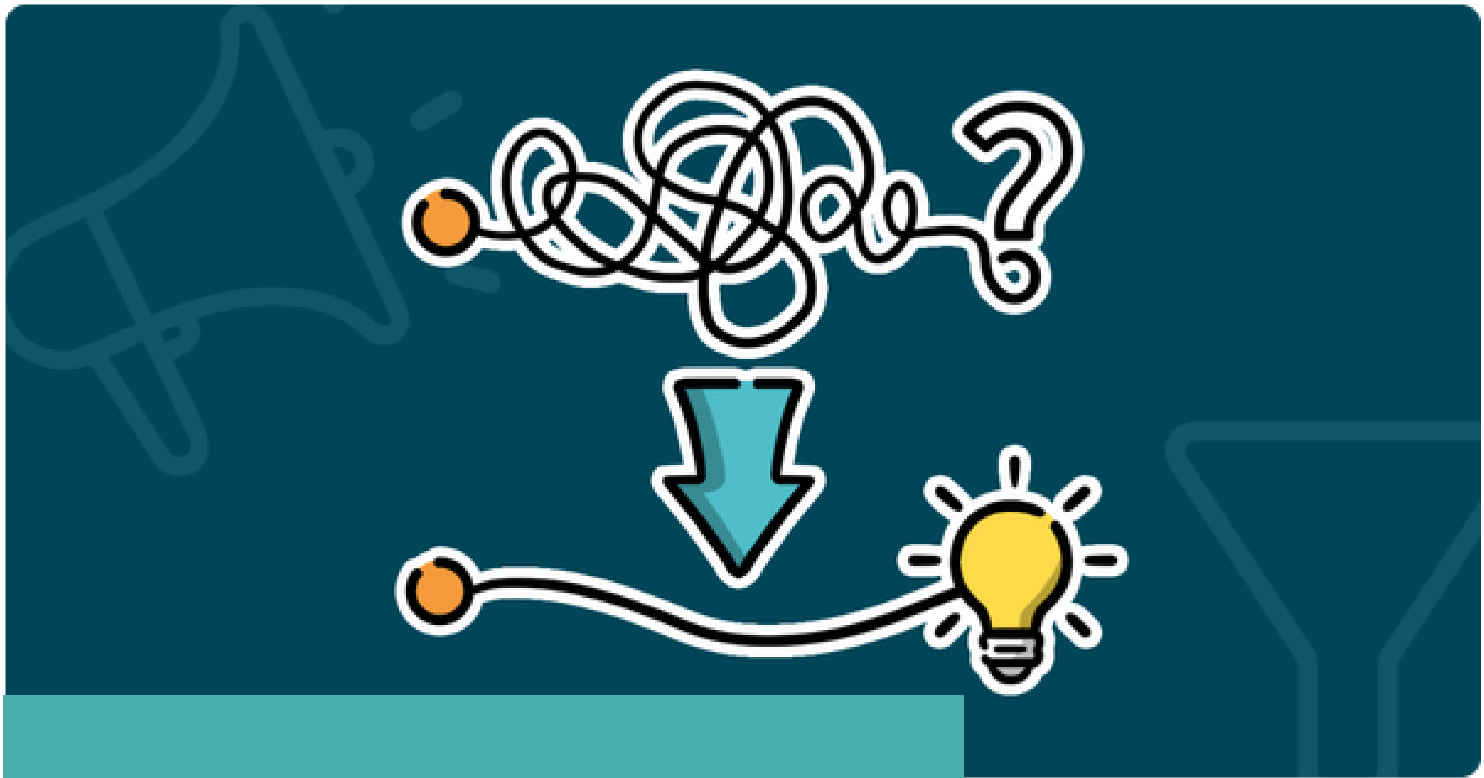
SUMMARIZED BY NOAM KIMHI

noam.kimhi@mail.huji.ac.il

# מודלים חישוביים, חישוביות וסיבוכיות

YEAR 2 | SEMESTER B | 2025



## סיכום קורס

COURSE NUMBER 67521

# תוכן עניינים

# שבוע 1 – הרצאה

## מבוא לחומר הקורס והגדרות מרכזיות

### משימות/בעיות חישוב

(1) למיין מערך נתון, $O(n\log n)$

(2) בהינתן גרף $G$ ושני צמתים $s,t$ למצוא מסלול קצר ביותר בין $s$ ל-$t$ (דוגמה לבעיה שבה כמה פלטים אפשריים), $\sim O(n^4)$

(3) בהינתן קליקה (קבוצת צמתים שמחוברים בקשת כולם אלו לאלו), מהו גודל הקליקה המקסימלית? $O(2^n\cdot n^2)$.

(4) בהינתן תוכנית פייתון, יש להכריע האם היא עוצרת. אי אפשר לפתור את השאלה הזו, זה ממיר את השאלה מתחום הסיבוכיות לתחום החישוביות.

המודל החישובי **משנה**. ראינו ב-DAST שמיון מבוסס השוואות מבוצע ב-$O(n\log n)$. באותה מידה, יכולנו להשתמש ב**מודל המקרונים**[1], שם אפשר לבצע את המיון ב-$O(n)$. האם זה מודל חישובי סביר? מה נחשב למודל חישובי סביר?

### בעיה חישובית

**הגדרה – א"ב:** קבוצה סופית, לא ריקה. $\boldsymbol{\Sigma}$.

**הגדרה – מילה:** מילה מעל א"ב $\boldsymbol{\Sigma}$ היא מחרוזת של אותיות מתוך $\boldsymbol{\Sigma}$.

מילה ללא אותיות כלל היא חוקית, ותסומן על ידי $\epsilon$. האות $\epsilon$ לא תיכלל ב-$\Sigma$ כדי למנוע בלבול.

**הגדרה – בעיה חישובית:** מורכבת מא"ב קלט $\boldsymbol{\Sigma}$, א"ב פלט $\boldsymbol{\Gamma}$ ופונקציה שמתאימה לכל מילה בקלט קבוצה כלשהי של מילות פלט ("הפלטים החוקיים עבור הקלט").

### בעיות הכרעה

**הגדרה – בעיית הכרעה:** בעיה חישובית שבה א"ב הפלט הוא מהצורה הבאה:

$$\Gamma=(\{T,F\},\{acc,rej\},\{Yes,No\})$$

ושבה לכל קלט הבעיה מתאימה קבוצת פלטים חוקיים המכילה מילה אחת בלבד, והמילה תהיה באורך אות אחת.

(בקצרה, לכל קלט התשובה צריכה להיות או כן – או לא).

[1] מודל המקרונים – מודל שבו אם עלינו למיין כמות מסוימת של מספרים, ניקח את אותה הכמות ונשבור מקרוני(ם) לפי הגודל שלהם. נארוז בכף היד וניישר, בכל פעם נוציא את המקרוני הארוך ביותר, וב-$n$-פעולות נוכל למיין את המערך.

דוגמה לבעיית הכרעה:

בהינתן גרף $G$, צמתים $s,t$ ומספר $k$, יש להחזיר "כן" אם יש מסלול באורך קטן/שווה $k$ מ-$s$ ל-$t$ בגרף $G$.

כמעט כל בעיית אופטימיזציה טבעית אפשר להמיר באמצעות מספר שינויים לבעיית הכרעה.

מה נוכיח בעתיד?

- אין תוכנית פייתון שפותרת את בעיית העצירה.
- אין אף מודל חישובי ריאליסטי (כזה שיש תקווה ליצור) שפותר את בעיית העצירה.
- יש בעיית הכרעה שאינה פתירה בעזרת תוכנית פייתון.

**הגדרה – שפה:** קבוצה של מילים.

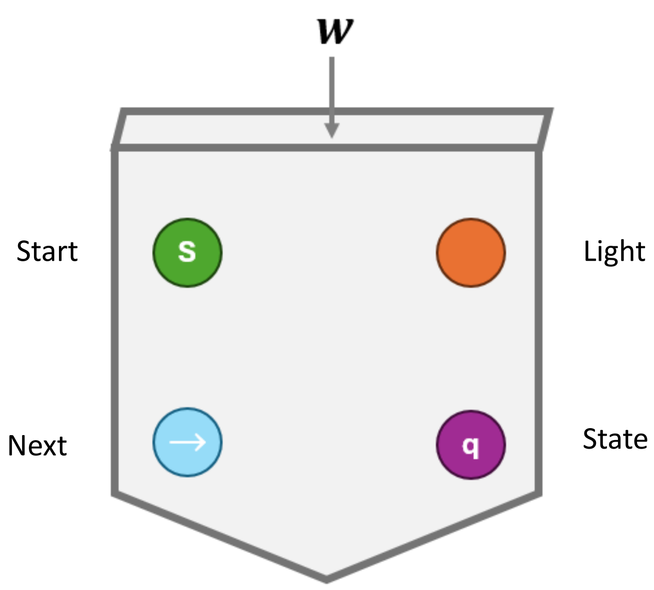
אבחנה: יש התאמה טבעית חח"ע ועל בין בעיות הכרעה מעל $\Sigma$ לבין שפות מעל $\Sigma$. נתאים לבעיית הכרעה המוגדרת בעזרת פונקציה $f$ את השפה $L_f$ המכילה את המילים מעל $\Sigma$ שעבורן $f$ מחזירה "$acc$".

ספירה

- $\Sigma$ – א"ב. מספר המילים באורך $k$ מעל $\Sigma$ הוא: $1 \leq |\Sigma|^k < \infty$.
- מספר המילים הכולל מעל $\Sigma$ הוא $\aleph_0$.
- מספר השפות מעל $\Sigma$ הוא: $\aleph = 2^{\aleph_0} > \aleph_0$. יש $\aleph_0$ תוכניות פייתון. יש $\aleph$ בעיות הכרעה מעל $\Sigma = \{a,b\}$, ומתקיים $\aleph > \aleph_0$ ולכן קיימות בעיות הכרעה מעל $\Sigma$ שלא ניתנות לפתרון בעזרת תוכניות פייתון. (למה יש $\aleph$ בעיות הכרעה? ראינו שיש $\aleph$ שפות והתאמה חח"ע ועל בין בעיות הכרעה לשפות).

# אוטומט סופי דטרמיניסטי – Deterministic Finite Automaton

תרשים להמחשה



נניח $\Sigma = \{a, b\}$ ומילה $aba$. נקרא משמאל לימין:

$$S \Rightarrow q_0 \Rightarrow a \rightarrow q_1 \Rightarrow b \rightarrow q_2 \Rightarrow a \rightarrow q_3$$

לבסוף, או שנדלקת הנורה אם המכונה קיבלה את הקלט, או שהיא לא נדלקת אם הקלט לא התקבל.

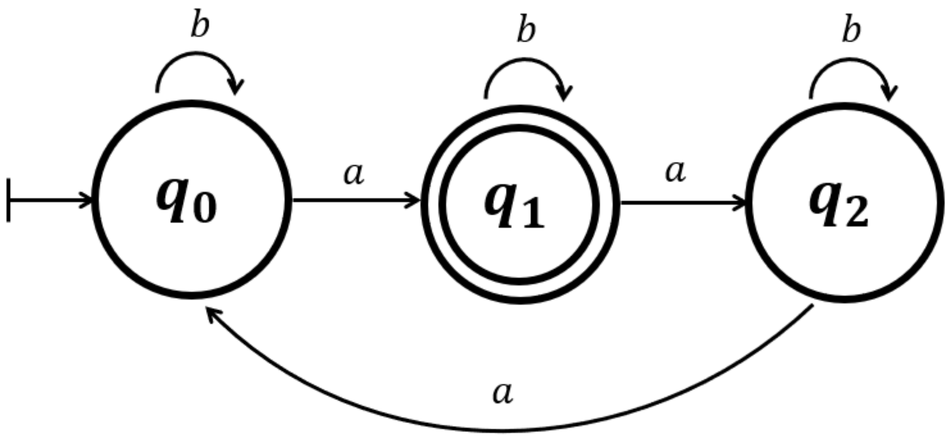
דוגמה:

$\Sigma = \{a, b\}$ ובעיית הכרעה: יש לקבל מילה $w$ אם"ם $\#_a(w) = 1 \ (mod\ 3)$.

למשל נקבל את המילים: $abaaa, bbbba$ ונדחה את המילה: $ababab$.

הגדרה לשפה המתאימה: $L = \{w\ over\ \{a, b\} \mid \#_a(w) = 1\ mod\ 3\}$

אוטומט מתאים לבעיה:




כאשר עיגול כפול הוא סימון למצב מקבל, וחץ עם קו אנכי הוא מצב התחלתי.

נסמן את האוטומט ב-$\mathcal{A}$. נגדיר את אוסף המילים שהאוטומט מקבל ב-$L(\mathcal{A})$.

**<u>הגדרה – אוטומט מכריע שפה:</u>** נאמר שהאוטומט $\boldsymbol{\mathcal{A}}$ מכריע את השפה $\boldsymbol{L}$ כאשר מתקיים $\boldsymbol{L(\mathcal{A}) = L}$.

הוכחה שהאוטומט המצויר בתרשים מכריע את $L = \{w\ over\ \{a, b\} \mid \#_a(w) = 1\ mod\ 3\}$:

נראה כי לכל מילה $w$ שיש בה $i$ מופעים של $a$ מודולו 3, האוטומט יגיע בריצתו עליה למצב $q_i$ (זו טענה חזקה יותר ממה שנדרשנו להוכיח). נוכיח באינדוקציה על אורך המילה $w$ שנסמן באות $n$.

<u>בסיס</u>: $n = 0$. עבור $w = \epsilon$ האוטומט מגיע למצב ההתחלתי $q_0$ ובזה מסיים בדיוק כפי שצריך.

<u>צעד האינדוקציה</u>: תהי $w$ מילה באורך $n + 1$ ונכתוב את $w$ בתור $w = w'\alpha$ כאשר $w'$ מילה באורך $n$ ו-$\alpha$ אות מבין $\{a, b\}$.

על $w'$ האוטומט מגיע למצב $q_i$ כאשר $i = \#_a(w')\ (mod\ 3)$, מהנחת האינדוקציה.

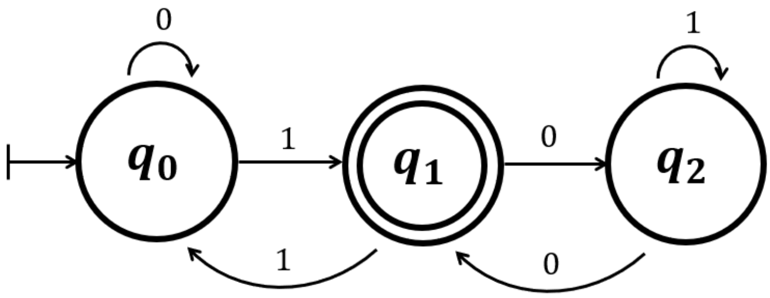
אם $\alpha = b$, האוטומט יישאר ב-$q_i$ וזה תואם לטענה.

אחרת $\alpha = a$, האוטומט יעבור מ-$q_i$ ל-$q_j$ כאשר $j = i + 1\ (mod\ 3)$ ואכן $\#_a(w) = i + 1\ (mod\ 3)$.

דוגמה:

א"ב $\Sigma = \{0,1\}$. שפה $L = \{w\ over\ \Sigma \mid (w)_2 = 1\ mod\ 3\}$ (כאשר זהו הסימון לצורתו הבינארית של המספר $w$).

הרעיון: בריצתו על מילה $w$ האוטומט יגיע למצב $q_i$ כאשר $i = (w)_2\ (mod\ 3)$ למשל 110 יתאים ל-$q_0$.




אפשר לחשוב על הבעיה בקריאה מימין לשמאל, ובכל הוספה של ספרה 0 או 1 לחשוב מה יקרה למספר שהיה שייך ל-$q_i$ לפני ההוספה.

## שאלות על אוטומטים

(1) האם ניתן להכריע את שפת הייצוגים הבינאריים של מספרים ראשוניים? לא.

(2) האם יש אפיון כללי פשוט ומעניין לאוסף הבעיות שניתן להכריע בעזרת אוטומט? כן.

(3) אם ניתן להכריע את השפה $L$ ואת השפה $L'$ בעזרת $DFA$ –
מה ניתן להגיד על $L \cap L'$? $L \cup L'$? $L \circ L'$?

**הגדרה – אוטומט סופי דטרמיניסטי DFA:** אוטומט $\boldsymbol{A}$ הוא החמישייה $\boldsymbol{A}\langle \boldsymbol{\Sigma}, \mathbf{Q}, \boldsymbol{q_0}, \boldsymbol{F}, \boldsymbol{\delta} \rangle$ כאשר:

$\Sigma$ – א"ב.

$Q$ – קבוצה סופית לא ריקה של מצבים.

$q_0$ – מצב התחלתי.

$F \subseteq Q$ – קבוצת מצבים מקבלים.

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

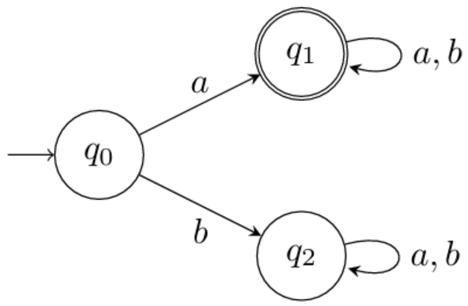
**הגדרה – ריצה של אוטומט:** **הריצה** של $\boldsymbol{A}$ על מילה $\boldsymbol{w} = \boldsymbol{\alpha_1\alpha_2 \dots \alpha_n}$ היא סדרת המצבים $\boldsymbol{q_0, \dots, q_n}$ שמקיימת $\boldsymbol{q_0}$ המצב ההתחלתי, $\boldsymbol{q_i = \delta(q_{i-1}, \alpha_i)}$.

**הגדרה – פונקציית המעברים המורחבת של $A$:** מוגדרת להיות $\delta^*(q_0, w) = q_n$ כאשר $w$ היא מילה מעל הא"ב, ומחזירה את המצב שהאוטומט מגיע אליו לאחר מעבר על כל $n$ האותיות במילה $w$.

# שבוע 1 – תרגול

## הוכחה שאוטומט מכריע שפה

הוצגה הוכחה שהאוטומט הבא:




מכריע את השפה $L = \{w \in \Sigma^* \mid w\ starts\ with\ 'a'\}$

<u>טענה</u>: $L(A) = L$

נוכיח טענת עזר: לכל מצב המילים הבאות מסיימות את ריצתן במצב:

$q_0$ – המילה הריקה.

$q_1$ – מילים המתחילות באות $a$

$q_2$ – מילים לא ריקות או שאינן מתחילות ב-$a$

$w \in L \iff$ $w$ מתחילה באות $a$ $\iff$ $A$ מסיימת ריצה על $w$ במצב $q_1$ $\iff$ $A$ מקבל את $w$ $\iff w \in L(A)$.

נוכיח את טענת העזר באינדוקציה על אורך המילה $w$:

<u>בסיס</u>: עבור $|w| = 0$ זו המילה הריקה, ההרצה על $w$ באוטומט נשארת במצב $q_0$.

נניח עבור כל מילה באורך $n$ ונוכיח עבור $w \in \Sigma^*$ כך ש-$|w| = n + 1$.

<u>צעד</u>: $q_0$ - $w'$ היא המילה הריקה ונקבל $\delta(q_0, w'\sigma) = \delta(q_0, \sigma)$.

אם $\sigma = a$ אז $\delta_0(q_0, a) = q_1$ ומאחר ש-$w = a$ בפרט מתחיל באות $a$, ואכן סיים ב-$q_1$.

אם $\sigma = b$ אז $\delta_0(q_0, a) = q_2$ ומאחר ש-$w = b$ בפרט לא מתחילה באות $a$, ואכן סיים ב-$q_2$.

$q_1$ - $w'$ התחילה באות $a$ מהנחת האינדוקציה. $\delta(q_1, \sigma) = q_1$, בפרט גם $w$ מתחילה ב-$a$ ואכן $A$ סיים את הריצה על $w$ ב-$q_1$.

את $q_2$ מראים באופן דומה.

**הגדרה פורמלית של פונקציית מעברים מורחבת**

$$\delta^*(q,w) \overset{\text{def}}{=} \begin{cases} q & w=\epsilon \\ \delta(\delta^*(q,w'),\sigma) & w=w'\sigma \end{cases}$$

**סימון – שרשור מילים:** יהיו $\boldsymbol{w_1, w_2 \in \Sigma^*}$ מילים מעל א"ב $\boldsymbol{\Sigma}$. שרשור שלהן מוגדר להיות $\boldsymbol{w_1 \cdot w_2}$.

טענה: יהי אוטומט $A\langle Q,\Sigma,\delta,q_0,F\rangle$. לכל $q\in Q$, $w,w'\in\Sigma^*$ מתקיים:

$$\delta^*(q,ww') = \delta^*(\delta^*(q,w),w')$$

הוכחה: באינדוקציה על $|w'|$

בסיס: $|w'|=0$ כלומר $w'=\epsilon$.

$$\delta^*(\delta^*(q,w),w') = \delta^*(\delta^*(q,w),\epsilon) = \delta^*(q,w) = \delta^*(q,w\epsilon) = \delta^*(q,ww')$$

הנחה: נניח נכונות עבור כל $|w'|=n$. נוכיח עבור $|w'|=n+1$.

צעד: נגדיר $w'=\overline{w}\sigma$ כאשר $|\overline{w}|=n$ ו-$\sigma\in\Sigma$.

$$\delta^*(q,ww') = \delta^*(q,w\overline{w}\sigma) = \delta(\delta^*(q,w\overline{w}),\sigma) = \delta(\delta^*(\delta^*(q,w),\overline{w}),\sigma) =$$
$$= \delta^*(\delta^*(q,w),\overline{w}\sigma) = \delta^*(\delta^*(q,w),w')$$

**הגדרה – שפה רגולרית:** שפה $\boldsymbol{L}$ תקרא רגולרית אם קיים אוטומט סופי דטרמיניסטי כך ש-$\boldsymbol{L(A)=L}$.

# שבוע 2 – הרצאה

## שפות רגולריות

### שאלות על $REG$

(1) האם $L_1, L_2 \in REG$ גורר שגם $L_1 \cup L_2 \in REG$? $L_1 \cap L_2 \in REG$? $\overline{L_1} \in REG$?

(2) האם אפשר לקבל אפיון מלא לאוסף השפות הרגולריות?

### חזרה על $\delta$ ועל $\delta^*$

נגדיר אוטומט $A = \langle Q, \Sigma, q_0, F, \delta \rangle$. $\delta(q, \alpha)$ הוא המצב אליו נעבור מ-$q$ כאשר קוראים את $\alpha$.

נגדיר $w = w'\alpha$, אז $\delta^*(q, w) = \delta(\delta^*(q, w'), \alpha)$ אם $w \neq \epsilon$ ואחרת $\delta^*(q, \epsilon) = q$.

$\delta^*(q, w)$ – המצב שאליו נגיע אם נתחיל ב-$q$ ונעבור על כל האותיות ב-$w$.

למה $\delta^*$ שימושי? בין היתר להגדרות כמו - $L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$ (מקצר כתיבה).

### איחוד וחיתוך של שפות רגולריות

<u>טענה</u>: בהינתן $L_1, L_2 \in REG$ אז גם $L_1 \cup L_2 \in REG$ (ונסיק במקביל $L_1 \cap L_2 \in REG$).

<u>הוכחה</u>: בה"כ ניתן להניח כי $L_1$ ו-$L_2$ מעל אותו א"ב $\Sigma$.

מדוע ניתן להניח זאת? אם $L_1$ מעל $\Sigma_1$ ו-$L_2$ מעל $\Sigma_2$ נוכל להגדיר $\Sigma = \Sigma_1 \cup \Sigma_2$. עכשיו צריך להסביר למה $L_1$ ו-$L_2$ עדיין שפות רגולריות גם מעל $\Sigma$. צריך להגדיר אוטומט חדש שבו יש מצב "בור" (sink) שלא מקבל, ואם הכנסנו אות שזרה לא"ב המקורי של השפה, נשלח למצב הבור.

נסמן $A = \langle Q, \Sigma, q_0, F, \delta \rangle$, $L(A) = L_1$, $B = \langle P, \Sigma, p_0, G, \eta \rangle$, $L(B) = L_2$.

<u>הרעיון</u>: נבנה אוטומט עם קבוצת מצבים $Q \times P$. על מילה $w$ האוטומט יגיע למצב $\left(\delta^*(q_0, w), \eta^*(p_0, w)\right)$. האוטומט שנבנה הוא:

$$C = \langle Q \times P, \Sigma, (q_0, p_0), F \times P \cup Q \times G, \psi \rangle$$

מספיק להוכיח שהרעיון מתקיים כדי לסיים. זה יראה שכשאנחנו מריצים את האוטומט על מילה $w$ זה ייתן את המצבים שלהם בסוף הריצה, ולפי האופן בו הגדרנו את קבוצת המצבים המקבלים. לכן רק נגדיר את $\psi$ על ידי: $\psi\left((q, p), \alpha\right) = \left(\delta(q, \alpha), \eta(p, \alpha)\right)$. נסיים בכך שנראה שלכל מילה $w$ מתקיים:

$$\psi^*\left((q_0, p_0), w\right) = \left(\delta^*(q_0, w), \eta^*(p_0, w)\right)$$

נוכיח באינדוקציה על $|w| = n$.

<u>בסיס:</u> $n = 0$ ולכן $w = \epsilon$.

$$\psi^*\big((q_0, p_0), \epsilon\big) = (q_0, p_0) \overset{\text{הגדרה}}{=} \big(\delta^*(q_0, \epsilon), \eta^*(p_0, \epsilon)\big)$$

<u>הנחה:</u> נניח את נכונות הטענה עבור $|w| = n$.

<u>צעד:</u> נוכיח עבור $|w| = n + 1$.

$$\psi^*\big((q_0, p_0), w\big) = \psi^*\big((q_0, p_0), w'\alpha\big) \overset{\text{def}\,\psi}{=} \psi\big(\psi^*\big((q_0, p_0), w'\big), \alpha\big) \overset{I.H}{=}$$

$$= \psi\Big(\big(\delta^*(q_0, w'), \eta^*(p_0, w')\big), \alpha\Big) \overset{\text{def}\,\psi}{=} \psi\big(\delta(\delta^*(q_0, w'), \alpha), \eta(\eta^*(p_0, w'), \alpha)\big) =$$

$$= \psi\big(\delta^*(q_0, w'\alpha), \eta^*(p_0, w'\alpha)\big) \overset{\text{def}\,w}{=} \psi\big(\delta^*(q_0, w), \eta^*(p_0, w)\big)$$

זה נקרא **אוטומט מכפלה** בגלל העובדה שיצרנו את קבוצת המצבים המקבלים שלו עם מכפלה קרטזית.

## שרשור שפות

<u>**הגדרה – שרשור שפות:**</u> $\boldsymbol{L_1, L_2}$ שפות מעל $\boldsymbol{\Sigma}$ (זה אינו תנאי מחייב). נגדיר את השרשור של השפות להיות $\boldsymbol{L_1 \circ L_2 = \{wz \mid w \in L_1, z \in L_2\}}$.

מעט דוגמאות וסימונים:

$$\{\epsilon\} \circ L_1 = L_1 \circ \{\epsilon\} = L_1$$

$$L_1^0 = \{\epsilon\}$$

$$L_1^k = \overbrace{L_1 \circ \ldots \circ L_1}^{k\ times}$$

$$L^m \circ L^k = L^{m+k}$$

$$L \circ \emptyset = \emptyset$$

<u>**הגדרה – סגור קליני של שפה:**</u> סגור קליני של שפה $\boldsymbol{L}$ יסומן $\boldsymbol{L^*}$ ומוגדר להיות $\boldsymbol{L^* = \bigcup_{k=0}^{\infty} L^k}$

תרגיל:

(1) $L = \{a^n \mid n\ is\ even\}$ אז $L^* = L$. כל מילה מ-$L$ שייכת גם ל-$L^*$. אם נשרשר כמה מילים של $L$, מאחר שהאורך של כולן זוגי אז גם השרשור שלהן באורך זוגי.

(2) $L = \{a\}$ אז $L^* = \{a^n \mid n \geq 0\}$.

(3) $L = \{a, b\}$ אז $L^* = \{a, b\}^*$.

## יצירת שפות רגולריות

<u>טענה – פעולות רגולריות:</u>

(1) $\emptyset, \{\epsilon\}, \{a\} \in REG$

(2) $REG$ סגור לאיחוד, שרשור וסגור קליני.

(3) כל שפה ב-$REG$ ניתן לקבל משפות מסוג (1) בעזרת פעולות מסוג (2).

<u>דוגמה:</u>

$$L = \{w \mid the\ sequence\ aaa\ is\ in\ w\}$$

ניצור **ביטוי רגולרי**:

$$L = \underbrace{\underbrace{\underbrace{(\{a\} \cup \{b\})^*}_{words\ in\ \{a,b\}} \circ \underbrace{\{a\} \circ \{a\} \circ \{a\}}_{\substack{the\ language\ with\\ aaa}}}_{words\ in\ \{a,b\}\ that\ end\ with\ aaa} \circ (\{a\} \cup \{b\})^*}_{words\ in\ \{a,b\}\ with\ the\ sequence\ aaa\ in\ them}$$

## מבט קדימה

כעת נגדיר "מודל חישובי" חדש NFA, נגדיר את NREG (כל השפות שמזוהות על ידי NFA), נראה ש-NREG סגור לשרשור ולסגור קליני ואז נסיים בכך שנראה REG=NREG.

# אוטומט סופי לא דטרמיניסטי – Non Deterministic Finite Automaton

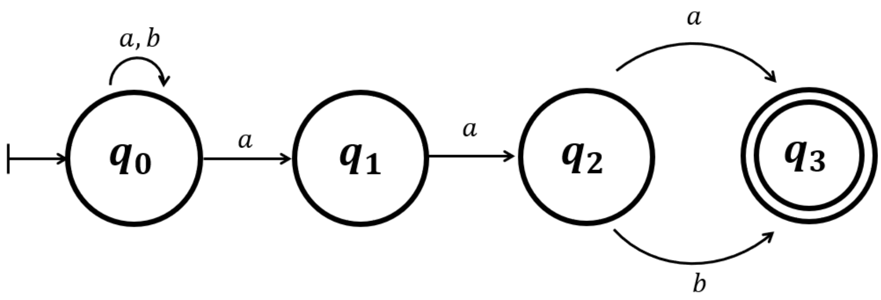
## NFA

בהינתן מילה, יש לאוטומט כזה קבוצה של ריצות חוקיות (תתכן ריצה אחת חוקית, מספר ריצות, או קבוצה ריקה). מגדירים שה-$NFA$ שנסמן ב-$A$ מקבל אות $w$ אם יש לו ריצה על $A$ שמסתיימת במצב מקבל.

<u>דוגמה:</u>

$$L_1 = \{w \in \{a,b\}^* \mid w\ ends\ with\ aaa\ or\ with\ aab\}$$

<u>הרעיון:</u> נצא מהמצב ההתחלתי רק על האות השלישית לפני הסוף.




נשים לב שב-$q_1$ וב-$q_2$ לא מוגדר מה קורה אם ניתקל באות $b$, ולכן זו תהיה ריצה לא חוקית אם נגיע למצב כזה.

דוגמת הרצה:

| | $a$ | | $b$ | | $a$ | | $a$ | | $b$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $q_0$ | | $q_0$ | | $q_0$ | | $q_0$ | | $q_0$ | | $q_0$ |
| $q_0$ | | $q_1$ | | X | | X | | X | | X |
| $q_0$ | | $q_0$ | | $q_0$ | | $q_1$ | | $q_2$ | | $q_3$ |

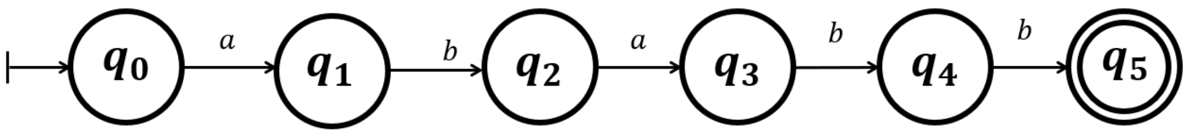
**הגדרה – אוטומט מזהה שפה:** אומרים על $\boldsymbol{NFA}$ שהוא מזהה את השפה $\boldsymbol{L(A)}$.

## איחוד שפות NREG

יהיו $L, L' \in NREG$. האם $L \cup L' \in NREG$? כן. נבנה אוטומט שמזהה את השפה $L \cup L'$ באמצעות חיבור ה-NFAs של השפות המקוריות.

דוגמה:

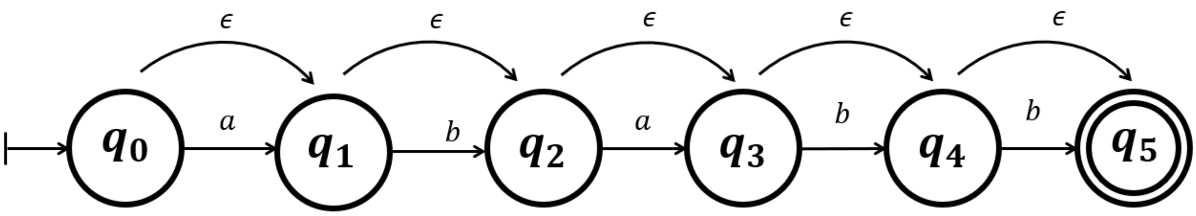
נתונה שפה $L_2 = \{ababb\}$ ונתון אוטומט לא דטרמיניסטי שמזהה אותה:




אם ננסה להריץ למשל את המילה $ababbb$ זו לא מילה שתקבל ריצה חוקית – וזה בסדר כי היא לא בשפה.

נגדיר: $L_3 = \{all\ the\ words\ received\ by\ erasing\ letters\ from\ 'ababb'\}$

איך ניצור אוטומט ל-$L_3$? נוסיף **מעברי $\boldsymbol{\epsilon}$** לאוטומט של $L_2$. מעברים אלו מאפשרים לנו להתחיל לא רק מ-$q_0$.




למשל עבור $bab \in L_3$ נדלג על $q_0$ ישר ל-$q_1$, משם נוכל או להגיע ל-$q_2$ עם ניצול האות $b$, או עם מעבר $\epsilon$ בלי ניצול של האות $b$.

## שבוע 2 – תרגול

### הגדרה פורמלית של $NFA$

**הגדרה – NFA**: $\boldsymbol{NFA}$ הוא חמישייה $\boldsymbol{A\langle Q,\Sigma,\delta,Q_0,F\rangle}$ כך ש-

$Q$ - קבוצת מצבים.

$\Sigma$ – א"ב.

$\delta: Q\times\Sigma\to 2^Q$ פונקציית מעברים.

$Q_0\subseteq Q$ קבוצת מצבים התחלתיים.

$F\subseteq Q$ – קבוצת מצבים מקבלים.

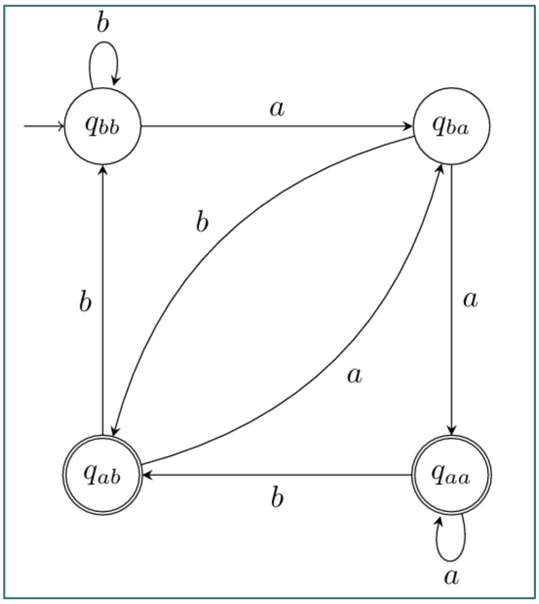
### ריצה של $NFA$

ריצה של NFA שנסמן ב-$A$ על מילה $w=w_1\dots w_n$ היא $r_0,r_1,\dots,r_n\in Q$ כך ש-$r_0\in Q_0$ וגם מתקיים $r_{i+1}\in\delta(r_i,w_{i+1})$ לכל $0\le i\le n$. ריצה מקבלת את $r_n\in F$. $A$ מזהה את $w$ אם <u>קיימת</u> ריצה מקבלת שלו עליה.

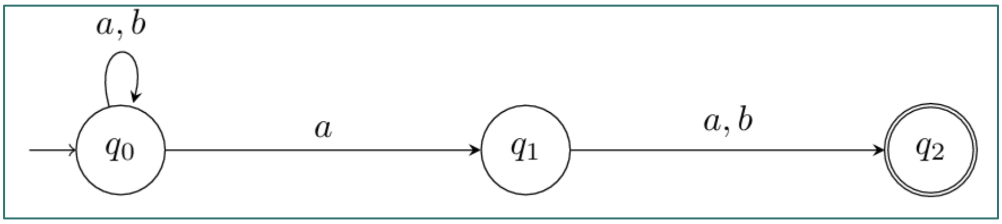
דוגמה:

$\Sigma=\{a,b\}$ ולכל $k\in\mathbb{N}$ נגדיר $L_k=\{w\in\Sigma^*\mid the\ k^{th}\ letter\ of\ w\ from\ the\ end\ is\ a\}$

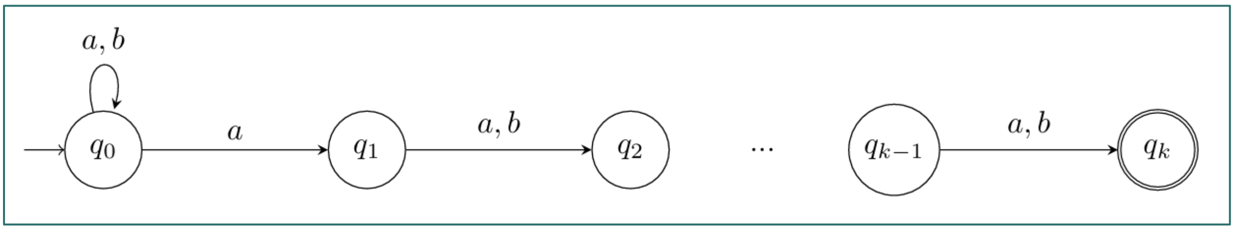
(1) *DFA* שמכריע את $L_2$:



(2) NFA שמזהה את $L_2$:




(3) עבור $k$ כללי, הראו שיש NFA עם $k+1$ מצבים שמזהה את $L_k$:




נרצה לאפיין לכל תת-קבוצה של מצבים אילו מילים $w$ יכולות לסיים את ריצתן ב-$Q'$. כל מילה יכולה לסיים ב-$q_0$, למילה $w$ נסמן $i_1, \dots, i_l$ את המיקומים של $a$ מהסוף רק ב-$k$-האותיות האחרונות, ונטען שהמצבים אליהם תגיע הם $\{q_0, q_{i_1}, \dots, q_{i_l}\}$.

(4) עבור $k$ כללי, הראו שכל DFA שמכריע את $L_k$ הוא בעל לפחות $2^k$ מצבים.

נניח בשלילה ש-$A$ עם פחות מ-$2^k$ מצבים כך ש-$L(A) = L_k$. נשים לב שיש $2^k$ מילים באורך $k$, לכן מעקרון שובך היונים קיימות 2 מילים $u \neq v$ שמסיימות את ריצותיהן באותו מצב. לכל $w \in \Sigma^*$ מתקיים כי $vw$ ו-$uw$ מסיימות באותו מצב ולכן הן מקבלות או נדחות ביחד. אם הן שונות באינדקס $i$ כלשהו (וחייב להיות כזה, אחרת זו אותה מילה והנחנו שהן שונות) אז נוסיף מילה באורך $i-1$, ואז אם $u[i] = a$ ו-$v[i] = b$ אז $uw \in L_k$ בעוד ש-$vw \notin L_k$ בסתירה.

## סגירות NREG לאיחוד

$L_1, L_2 \in NREG \Longleftarrow$ מתקיים $L_1 \cup L_2 \in NREG$ מעל $\Sigma$.

<u>הוכחה:</u>

נתון $L_1, L_2 \in NREG$ ולכן קיים $A = \langle Q, \Sigma, \delta, Q_0, F\rangle$ עבורו $L(A) = L_1$ וכן $B = \langle P, \Sigma, \eta, P_0, G\rangle$ עבורו $L(B) = L_2$.

נגדיר $C = \langle Q \cup P, \Sigma, \alpha, Q_0 \cup P_0, F \cup G\rangle$, כאשר $\alpha(Q,\sigma) = \begin{cases} \delta(q,\sigma) & q \in Q \\ \eta(q,\sigma) & q \in P \end{cases}$

תהי $w = w_1 \dots w_n$ ו-$r_0, \dots, r_n$ הריצה של $C$ על $w$. נראה שכל הריצה מוכלת או ב-$Q$ או ב-$P$.

עבור $r_0 \in Q_0$ אפשר להראות (באינדוקציה) $r_{i+1} \in \alpha(r_i, w_{i+1}) = \delta(r_i, w_{i+1}) \in Q$ ובאותו אופן אם $r_0 \in P_0$.

נוכיח ש-$L(C) = L_1 \cup L_2$ בהכלה דו כיוונית.

(1) תהי $w \in L_1 \cup L_2$, בה"כ $w \in L_1$. קיימת ריצה מקבלת של $A$ על $w$ ונסמנה $r$ ריצה מקבלת של $C$ על $w$.

(2) תהי $w \in L(C)$. קיימת ריצה מקבלת של $C$ על $w$. בה"כ $r_0 \in Q_0$, לכן הריצה $r$ מוכלת כולה ב-$Q$ וזו ריצה מקבלת גם שם, לכן $L(C) = L_1 \cup L_2$.

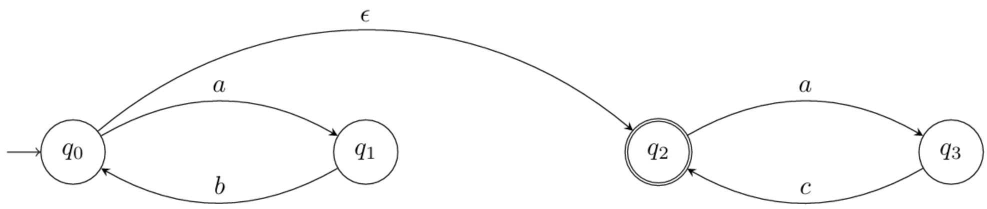
### סגירות NREG לשרשור

שפה NREG מקיימת סגירות לשרשור. באותם סימונים כמו קודם, נגדיר: $C\langle Q \cup P, \Sigma, \alpha, Q_0, G\rangle$ עם $\alpha$ פונקציית מעברים שהיא כל המעברים של $A$, כל המעברים של $B$ ומעבר אפסילון בין כל מצב מקבל ב-$A$ לכל מצב התחלתי ב-$B$.

דוגמה מהתרגול:

For example, consider $L_1 = (ab)^*$ and $L_2 = (ac)^*$ over $\Sigma = \{a, b, c\}$, and the following automata:


Let's apply the construction described above for this concrete example:




נרצה לתאר אוטומט ללא שימוש במעברי אפסילון, כיצד נשיג זאת? השינוי מאוטומט עם מעבר אפסילון לבלי הוא הוספת חץ התחלה למצב היעד של מעבר ה-$\epsilon$, ובדוגמה מאחר שכל מעבר ל-$q_0$ יכול היה להוביל גם ל-$q_2$ הוספנו מעבר מ-$q_1$ ל-$q_2$ באמצעות $b$.

<u>הכללה של הרעיון</u>: לכל מצב $q$ נגדיר $E(q) = \{q' \in Q \mid q' \text{ is reachable from } q \text{ using } \epsilon\}$ ואז $\delta(q, \sigma) = \cup_{q' \in \eta(q,\sigma)} E(q')$. נשנה גם את קבוצת המצבים ההתחלתיים $Q_0 = \cup_{q \in P_0} E(q)$.

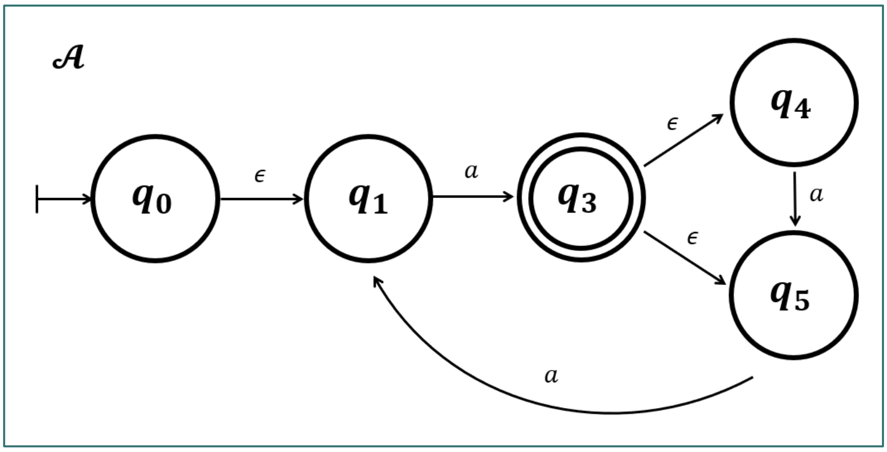
**<u>הגדרה – פונקציית מעברים מורחבת $\delta^*$ ב-$NFA$</u>**: פונקציית מעברים מורחבת מוגדרת באופן הבא:

$$\delta^*(S, w) = \begin{cases} S & w = \epsilon \\ \bigcup_{q \in \delta^*(S, w')} \delta(q, \sigma) & w = w' \cdot \sigma \end{cases}$$

# שבוע 3 – הרצאה

## מעברי אפסילון

<u>דוגמה:</u>




$$\mathcal{A} = \langle \Sigma, Q, Q_0, F, \delta \rangle$$

$Q_0 = \{q_0, q_1\}$ – בגלל מעבר ה-$\epsilon$ מ-$q_0$ ל-$q_1$, אפשר להתחיל גם ב-$q_1$.

$\delta: Q \times \Sigma \to 2^Q$ – מספר דוגמאות בה: $\delta(q_0, a) = \{q_4, q_3, q_1, q_5\}$, $\delta(q_5, a) = \{q_1\}$

<u>ריצה חוקית על $\boldsymbol{w \in \Sigma^*}$ ממצב $\boldsymbol{q \in Q}$</u>: עבור $\boldsymbol{w = w_1 w_2 \dots w_k}$, ריצה חוקית כנ"ל היא סדרת מצבים $\boldsymbol{q^0, q^1, \dots, q^k}$ אשר מקיימת $\boldsymbol{q^0 = q}$ ו-$\boldsymbol{q_i \in \delta(q^{i-1}, w_i)}$.

<u>ריצה על $\boldsymbol{w}$</u>: ריצה חוקית על $\boldsymbol{w}$ שמתחילה באיזשהו $\boldsymbol{q^0 \in Q}$.

<u>פונקציית המעברים המורחבת $\boldsymbol{\delta^*}$</u>: פונקציה $\boldsymbol{\delta^*: 2^Q \times \Sigma^* \to 2^Q}$, כאשר $\boldsymbol{\delta^*(S, w)}$ הוא:

(1) אוסף המצבים שניתן להגיע אליהם בריצה חוקית על $w$ שמתחילה במצב כלשהו מ-$S$.

(2) מוגדר להיות $S$ אם $w = \epsilon$, ואחרת אם $w = w'\alpha$ אז: $\bigcup_{q \in \delta^*(S,w)} \delta(q, \alpha)$.

<u>טענה</u>: הגדרות (1) ו-(2) שקולות לכל $w \in \Sigma^*$.

<u>הוכחה</u>: באינדוקציה על אורך המילה $|w|$, ראינו בעבר הוכחות דומות, בסיס באמצעות $w = \epsilon$, צעד באמצעות $w = w'\alpha$.

<u>דוגמה:</u>

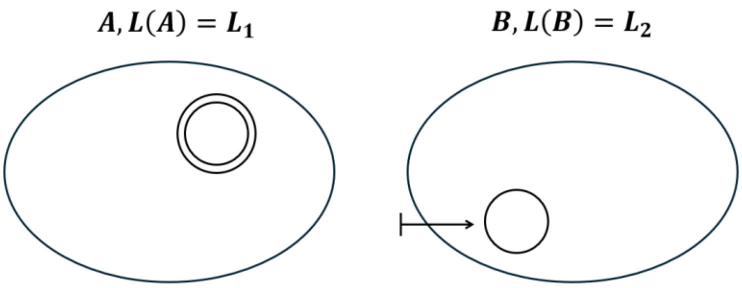
בחזרה לדוגמה עם האוטומט $\mathcal{A}$ לעיל, נקבל:

$$\delta^*(\{q_1, q_3\}, aa) = \{q_5, q_1, q_3, q_4\}$$

כאשר אל $q_5, q_1$ הגענו דרך $q_1$, ואל $q_1, q_3, q_4$ דרך $q_3$.

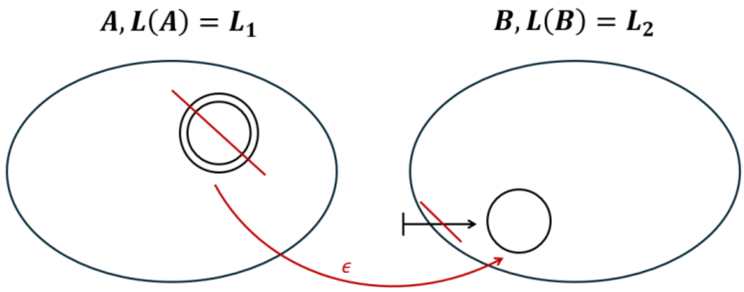
## יצירת אוטומט לשרשור שפות

נתונים ה-$NFAs$ (חלקיים) הבאים המגדירים את השפות:

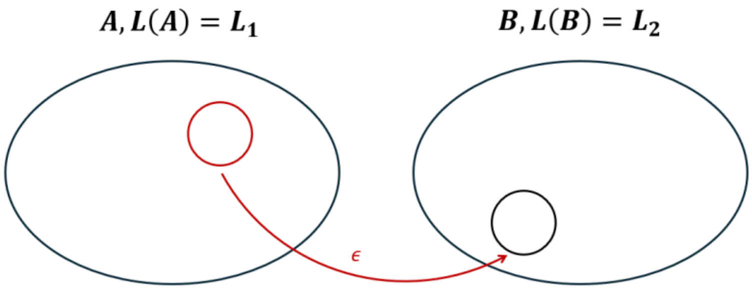



כיצד ניצור אוטומט לשרשור שלהם?

נוסיף מעברי $\epsilon$ מכל מצב מקבל ב-$A$ למצב התחלתי ב-$B$. נבטל את המצבים ההתחלתיים ב-$B$ להיות מצבים רגילים. נבטל את המצבים המקבלים ב-$A$ להיות מצבים רגילים:

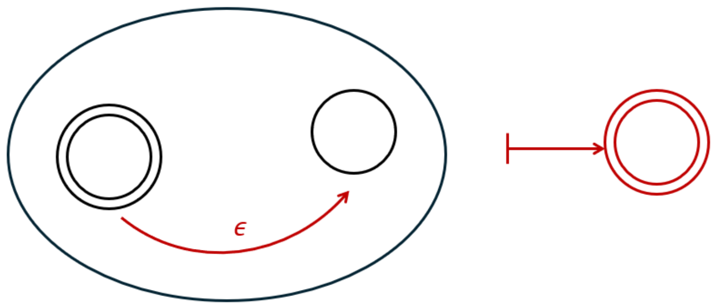



לבסוף נקבל:



## יצירת אוטומט לסגור קליני

נוסיף מעבר $\epsilon$ מכל מצב מקבל למצב התחלתי. נוסיף מצב התחלתי נוסף עבור מצבים של המילה הריקה.




## טענה מרכזית: $NREG = REG$

<u>הוכחה</u>: נוכיח באמצעות הכלה דו-כיוונית.

$REG \subseteq NREG$ – הוכח במסגרת תרגיל הבית.

$NREG \supseteq REG$ – תהא $L \in NREG$ ויהי $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ אוטומט לא דטרמיניסטי המכריע אותה.
נבנה אוטומט דטרמיניסטי $A_d$ המכריע את $L$ על ידי $A_d = \langle \Sigma, 2^Q, Q_0, F_d, \delta_d \rangle$.

<u>הרעיון</u>: ב-$A_d$ יהיה מצב עבור כל תת-קבוצה $S \subseteq Q$. בריצתו על מילה $w$, $A_d$ יגיע למצב $\delta^*(Q_0, w)$ (זו <u>קבוצת</u> מצבים). באופן פורמלי:

$F_d$ – קבוצת מצבים מקבלים על ידי: $F_d = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$.
$\delta_d$ – פונקציית מעברים המוגדרת על ידי: $\delta_d(S, \alpha) = \delta^*(S, \alpha)$.
<u>טענה</u>: $L(A_d) = L(A)$

מספיק להראות $\delta_d^*(Q_0, w) = \delta^*(Q_0, w)$ כי אז נקבל:

$$w \in L(A_d) \Leftrightarrow \delta^*(Q_0, w) \in F_d \Leftrightarrow \delta^*(Q_0, w) \in F_d \Leftrightarrow \delta^*(Q_0, w) \cap F \neq \emptyset \Leftrightarrow w \in F_d$$

נוכיח באינדוקציה על אורך המילה $|w|$.

<u>בסיס</u>: $|w| = \epsilon$. $\delta_d^*(Q_0, \epsilon) = Q_0 = \delta^*(Q_0, \epsilon)$

<u>צעד</u>: נסמן $w = w'\alpha$, נקבל:

$$\begin{aligned}\delta_d^*(Q_0, w'\alpha) &= \delta_d(\delta_d^*(Q_0, w'), \alpha) \overset{I.H}{=} \delta_d(\delta^*(Q_0, w'), \alpha) \overset{\text{def}\,\delta_d}{=} \delta^*(\delta^*(Q_0, w'), \alpha) \\ &= \delta^*(Q_0, w'\alpha) = \delta^*(Q_0, w)\end{aligned}$$

מסקנה: $NREG = REG$.

שאלה: האם הניפוח האקספוננציאלי של מספר המצבים הכרחי? **כן.**

דוגמה:

$\Sigma = \{0,1\}$ עבור $k \in \mathbb{N}$, נגדיר $L_k \coloneqq \Sigma^* \circ \{1\} \circ \Sigma^{k-1}$ (כלומר, האות ה-$k$ מהסוף היא 1).

## שקילות Myhill-Nerode

הוצגה ההגדרה הבאה:

**הגדרה – שקילות MN:** $\boldsymbol{L}$ שפה מעל $\boldsymbol{\Sigma}$, ו-$\boldsymbol{x, y \in \Sigma^*}$ (לא בהכרח בשפה/מחוצה לה). נגדיר ש-$\boldsymbol{x}$ ו-$\boldsymbol{y}$ **שקולות MN** ביחס לשפה $\boldsymbol{L}$ אם לא קיימת סיומת מפרידה $\boldsymbol{z \in \Sigma^*}$, כך ש-$\boldsymbol{xz \in L}$ ו-$\boldsymbol{yz \notin L}$, או $\boldsymbol{xz \notin L}$ ו-$\boldsymbol{yz \in L}$. במקרה זה נסמן כי: $\boldsymbol{x \sim_L y}$. אם קיימת סיומת מפרידה $\boldsymbol{z}$, אז $\boldsymbol{x \nsim_L y}$.

אבחנות:

(1) נניח ש-$\boldsymbol{x \sim_L y}$, ו-$x \in L$, אז $y \in L$ . תקף גם עם $\notin$.

(2) נניח ש-$\boldsymbol{x \nsim_L y}$ וש-$L = L(A)$, אז בריצתו על $x$ ועל $y$, $A$ מגיע למצבים שונים.

(3) אם יש $n$ מילים ב-$\Sigma^*$ שאינן שקולות MN, אז ל-DFA שמכריע את L יש לפחות $n$ מצבים. בניסוח אחר: אם ביחס $\sim_L$ יש $k$ מחלקות שקילות, אז ב-DFA שמכריע את L יש לפחות $k$ מצבים.

הערה: אם יודעים $\boldsymbol{x \sim_L y}$ אז מחלקות השקילות של מילים אלו שוות: $[x]_L = [y]_L$. למשל, $L = \{a, aa\}$, המילה $z = a$ סיומת מפרידה, כי $aaa \notin L$, $aa \in L$. מזה אפשר להסיק בעזרת (3) שיש לפחות 2 מצבים ב-DFA שמזהה את השפה $L$.

מסקנה:

אם יש $\infty$ מחלקות שקילות $\sim_L$, אז $L \notin REG$. אם ב-$\sim_L$ יש לפחות $k$ מחלקות, אז ב-DFA של $L$ יש לפחות $k$ מצבים, לכן אם יש אוטומט $A$ עם $n$ מצבים שמכריע את $L$ אז מספר המחלקות ב-$\sim_L$ לכל היותר $n$.

חזרה לדוגמה של $L_k$:

עבור כל שתי מילים שונות $x, y \in \{0,1\}^k$, $x \nsim_{L_k} y$ עבור מילים כאלו קיים $1 \le i \le k$ עבורו $x_i \neq y_i$ (אחרת היו זהות), נקבע את $z$ להיות $\{0\}^{k-i-1}$, ואז האות ה-$k$ מהסוף ב-$xz$ היא $x_i$ והאות ה-$k$ לפני הסוף ב-$yz$ היא $y_i$. $x_i \neq y_i$ ולכן רק אחד מבין $xz$ ו-$yz$ יהיו בשפה $L_k$.

מכאן נסיק – אוטומט שמכריע את $L_k$ הוא בעל לכל הפחות $2^k$ מצבים.

דוגמה: $\Sigma = \{a, b\}$, $L = \{a^n b^n \mid n \geq 0\}$, עבור כל $m \neq n$, $a^m \nsim_L a^n$. למה? נבחין כי $a^n b^n \in L$, לעומת $a^m b^n \notin L$. יש מספר אינסופי של מחלקות שקילות $L \notin REG$.

טענה: אם $L$ שפה מעל $\Sigma^*$ וב-$\sim_L$ יש $k < \infty$ מחלקות שקילות, אז $L \in REG$, ויש DFA עבור $L$ עם $k$ מצבים.

רעיון: נסמן את קבוצת מחלקות השקילות ביחס $\sim_L$ ב-$Q$. כמו כן, $\delta^*(q_0, w) = [w]_L$.

בניית $A$: $A = \langle \Sigma, Q, [\epsilon]_L, F, \delta \rangle$, כאשר:

$$F = \{[w]_L \mid w \in L\}$$

$$\delta([w]_L, \alpha) = [w\alpha]_L$$

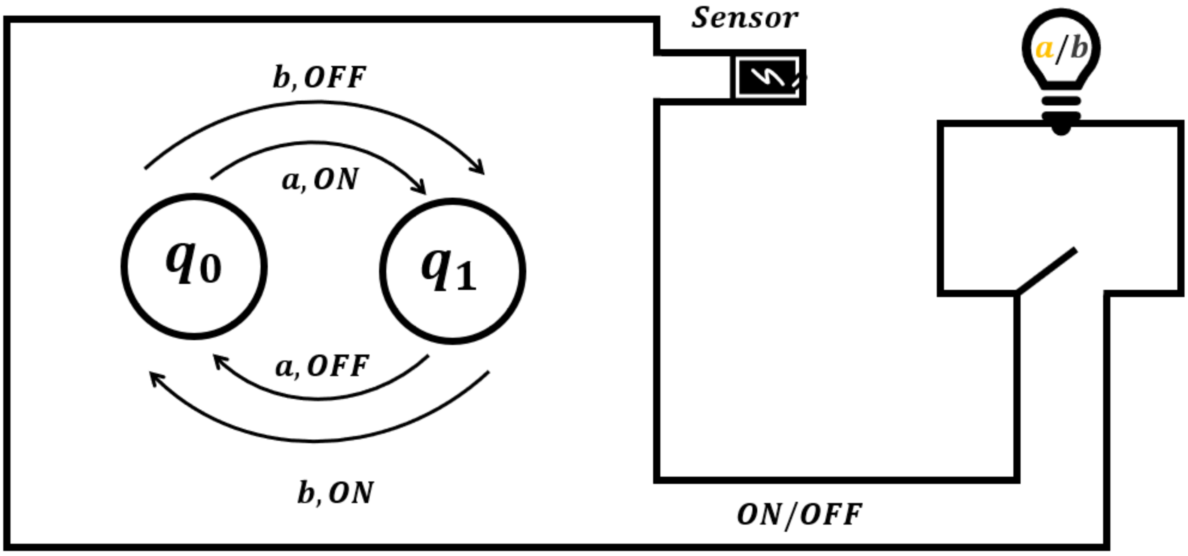
יש לוודא גם שאם $[w]_L = [y]_L$ אז גם $[w\alpha]_L = [y\alpha]_L$. נראה זאת:

$$[w]_L = [y]_L \Longrightarrow w \sim_L y \Longrightarrow \forall z \quad wz \in L \; iff \; yz \in L \overset{z=\alpha z'}{\Longrightarrow} \forall z' \quad w\alpha z' \in L \; iff \; y\alpha z' \in L \Longrightarrow$$
$$w\alpha \sim_L y\alpha \Longrightarrow [w\alpha]_L = [y\alpha]_L$$

# שבוע 4 – הרצאה

## מכונת טיורינג – Turing Machine (TM)

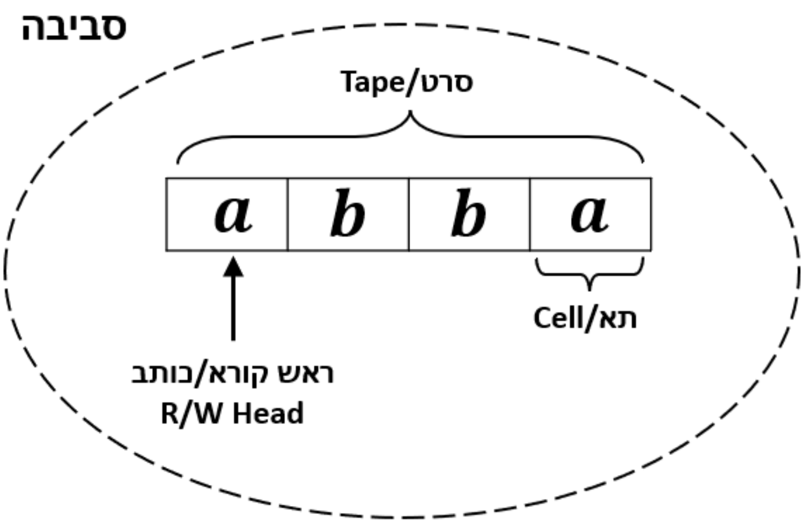
אוטומט בקרה




ב-$q_0$ אם נראה שהנורה דולקת נשאיר אותה דולקת. אם היא כבויה תישאר כבויה.

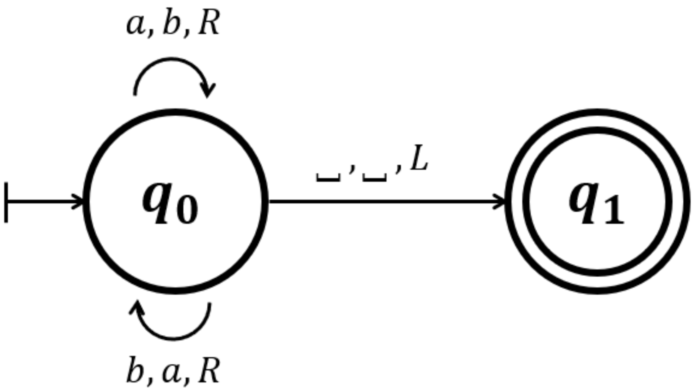
בפועל, הנורה תדלק ותכבה לסירוגין.

היכרות – מכונת טיורינג

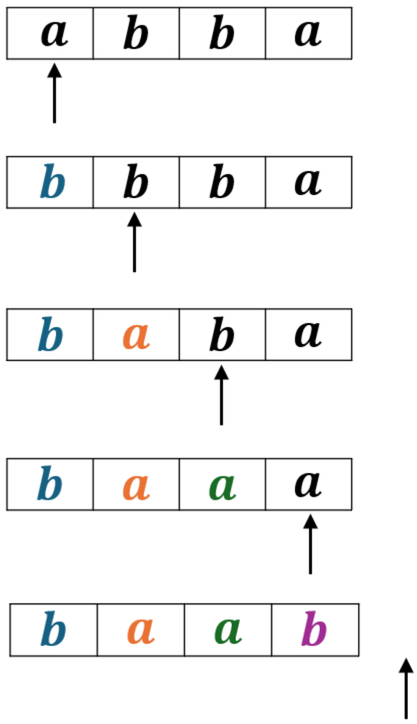


ראש קורא/כותב – מתחיל באות השמאלית ביותר של הקלט. שולח לאוטומט את האות שנמצאת מולו. על כל חץ האוטומט מוציא 2 הוראות: אות וכיוון.

<u>דוגמה:</u>

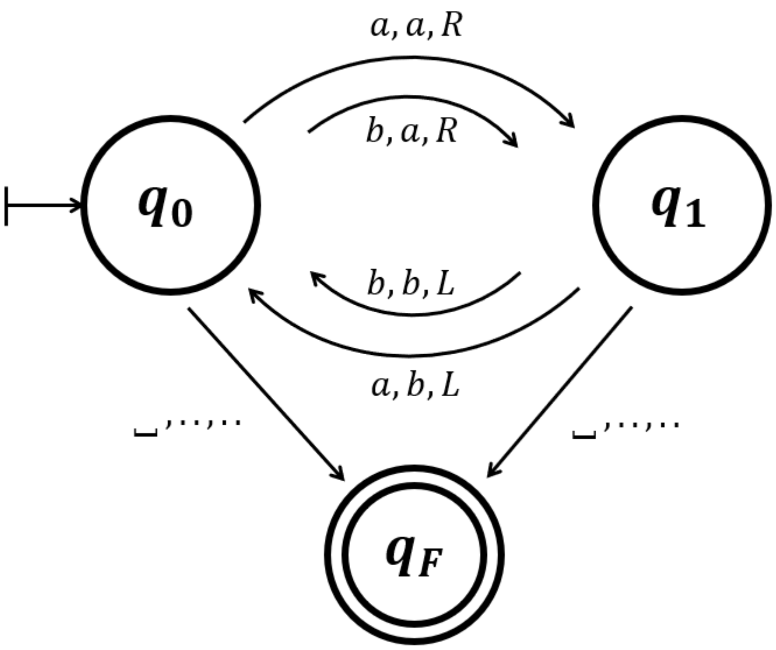



עבור הקלט $abba$ נקבל את הריצה הבאה:




כשמכונת טיורינג מגיעה אל מחוץ לסרט, הראש הקורא/כותב ממשיך אל מחוץ לה וכותב ␣ (רווח). משם נעבור ל-$q_1$ מצב סופי. אם למשל את החץ שיוצא ל-$q_1$ נשנה לחץ פנימי חזרה ל-$q_0$, המכונה תחליף כל $a$ ל-$b$ וכל $b$ ל-$a$, אבל הסרט לא יפסיק לגדול והמכונה תמשיך לכתוב רווחים ימינה.

דוגמה ל-TM שלא כותבת סרט שהולך וגדל, אבל גם לא עוצרת:




הגדרות

**קונפיגורציה** מה שכתוב על הסרט, מצב האוטומט, ומיקום הראש הכותב/קורא. לקונפיגורציה שבה הראש מכוון על האות הראשונה, המצב הוא המצב ההתחלתי ובסרט יש את הקלט נקרא קונפיגורציה התחלתית, ונסמנה $C_0$. הקונפיגורציה שתגיע אחרי $C_0$ תיקרא הקונפיגורציה העוקבת ל-$C_0$. נסמנה ב-$C_1$. לקונפיגורציה שבה המצב הוא מצב סופי נקרא קונפיגורציה סופית, ואין לה קונפיגורציה עוקבת. נסמן קונפיגורציה על מילה $\alpha_1\alpha_2 \ldots \alpha_i \ldots \alpha_k$ כאשר הראש מכוון על $\alpha_i$ והמ"ט במצב $q$ כך: $\alpha_1 \ldots \alpha_{i-1}q\alpha_i \ldots \alpha_k$.

**ריצה חלקית** של מ"ט T על קלט $w$ היא סדרה (סופית או אינסופית) של קונפיגורציות $C_0, C_1, C_2, \ldots$ כאשר $C_0$ היא הקונפיגורציה ההתחלתית בריצת T על $w$, ולכל $i > 0$ נקבל כי $C_i$ היא הקונפיגורציה העוקבת של $C_{i-1}$.

**ריצה מלאה** היא כאשר הריצה סופית ומסתיימת בקונפיגורציה סופית, או אם היא אינסופית.

**זמן ריצה** (של מ"ט T על קלט $w$) הוא אורך הריצה המלאה שלה על הקלט $w$, פחות 1. עבור ריצה אינסופית – אינסוף.

**קלט** – הסרט ההתחלתי.

**פלט** – תוכן הסרט בסוף הריצה המלאה (אם היא סופית). עבור ריצה אינסופית אין פלט.

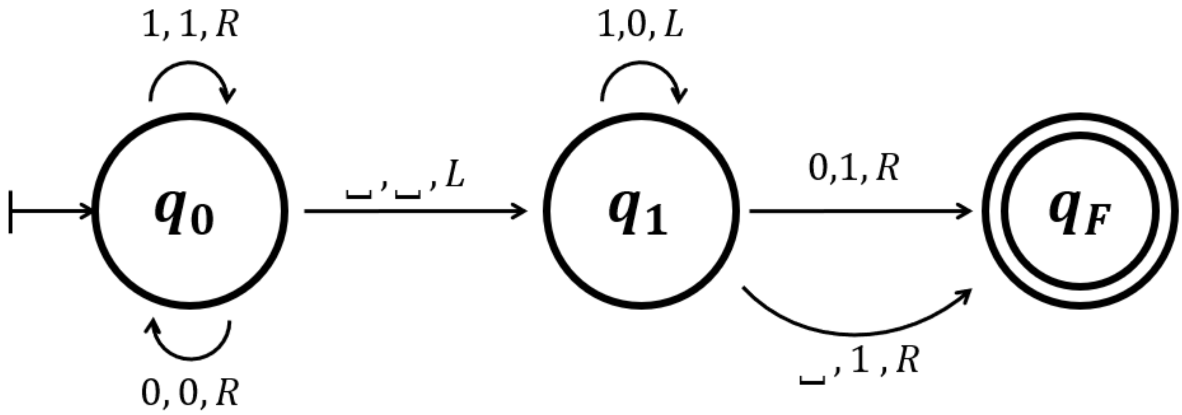
## בעיה לדוגמה – מספר עוקב

קלט: $x \in \{0,1\}^*$. פלט: $x+1$.

אלגוריתם:

(1) נחפש את הקצה הימני של הקלט.
(2) נהפוך כל 1 שנתקל בו ל-0, עד שנמצא 0, נהפוך אותו ל-1 ונסיים.

זמן ריצה: $2n+O(1)$, במקרה הגרוע ביותר בו לא נתקלנו ב-0 בכל הדרך.




## בעיה לדוגמה – מספר קודם

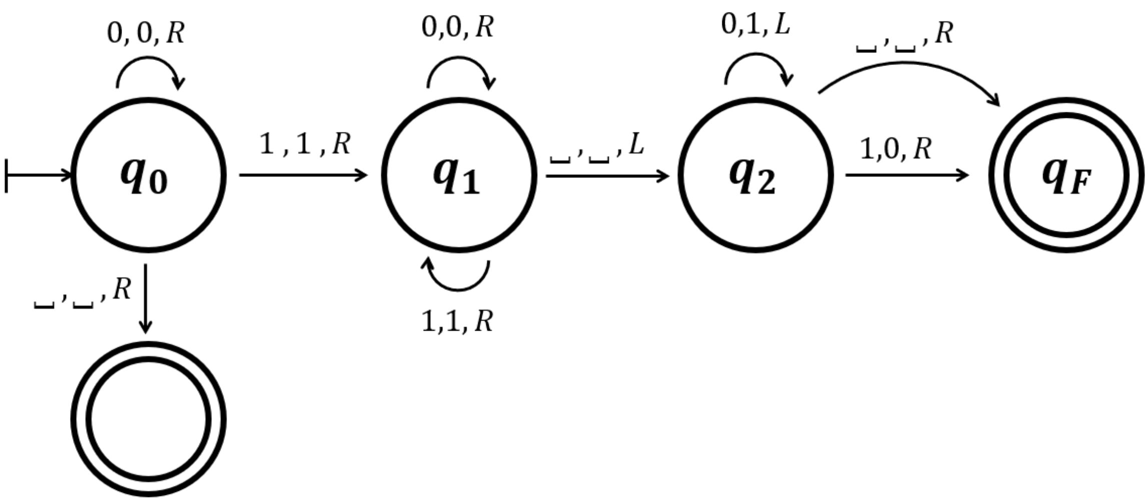
קלט: $x \in \{0,1\}^*$. פלט: $x-1$ אם $x>0$, אחרת 0.

אלגוריתם:

בגדול, הרעיון הוא לעבור על המספר מימין לשמאל, אפסים לשנות ל-1, ואת ה-1 הראשון שניתקל בו נהפוך ל-0. מזכיר את האוטומט הקודם פרט למקרה של $x \le 0$.

(1) נחפש את הקצה הימני של הקלט, אם הקלט הוא 0 נעצור.

(2) כמו קודם החל מהמצב $q_2$ בתרשים הבא:



**הגדרה פורמלית מכונת טיורינג** – מ"ט היא שביעייה $\langle \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \sqcup, \boldsymbol{Q}, \boldsymbol{q_0}, \boldsymbol{F}, \boldsymbol{\delta} \rangle$ כאשר:

$\Sigma$ – א"ב הקלט, קבוצה סופית לא ריקה.

$\Gamma$ – א"ב העבודה, יקיים $\Sigma \subseteq \Gamma$ (סופי).

$\sqcup$ - תו מיוחד, $\sqcup \in \Gamma \setminus \Sigma$.

$Q$ – קבוצת המצבים (סופית).

$q_0 \in Q$ – המצב ההתחלתי.

$F \subseteq Q$ – קבוצת המצבים הסופיים. לרוב ניתקל ב-$F = \{q_F\}$ או ב-$F = \{q_{acc}, q_{rej}\}$.

$\delta$ – פונקציית המעברים: $\delta: \Gamma \times (Q \setminus F) \to Q \times \Gamma \times \{R, L\}$.

## חיבור מספרים

<u>קלט</u>: $x\#y$, $x, y \in \{0,1\}^*$. <u>פלט</u>: $x + y\#$

<u>מימוש</u>:

(1) נלך ימינה עד תא אחד לפני התו # (נגיע לתחילת $y$).
(2) נחסר אחד מ-$y$, אם $y = 0$ נעבור על האותיות של $y$ מימין לשמאל ונהפוך אותך לרווחים. נעצור כאשר נגיע אל התו #.
(3) נלך שמאלה עד תא אחד שמאלה מ-#. כרגע המצב הוא: $x\#(y-1)$
(4) נוסיף 1 ל-$x$.
(5) נחזור לשלב 1.

<u>זמן ריצה</u>: אם $n$ הוא האורך של $x, y$ אז שלבים 1,2,3 לוקחים $O(n)$. חוזרים על כך $y$ פעמים, ובמקרה הגרוע ביותר $x$ הוא תו אחד, ולכן $|y| = n - 2$ ונקבל: $O(n) \cdot 2^{n-2}$.

## העתקת מחרוזת

<u>קלט</u>: $x \in \{a, b\}^*$ <u>פלט</u>: $x\#x$.

איך נדע אילו אותיות כבר העתקנו? באמצעות קונספט חדש שנקרא **סימון**.

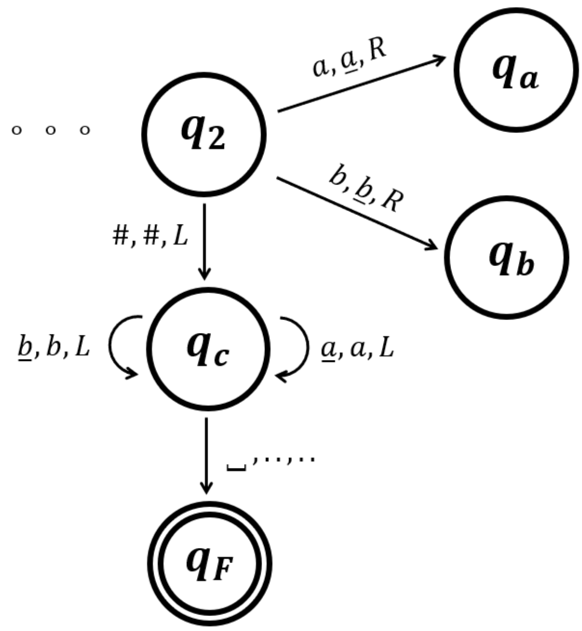
**סימון** – הגדרת א"ב עבודה חדש על ידי: $\boldsymbol{\Gamma} = \boldsymbol{\Sigma} \cup (\boldsymbol{\Sigma} \times \{\_, \bar{\ }\}) \cup \{\sqcup\} \cup \ldots$.

פורמלית מדובר על הזוג הסדור $(a, \_)$ אבל אנחנו נסמן $\underline{a}$.

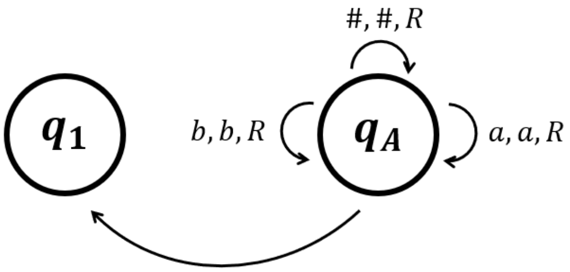
בדוגמה שלנו: $\Sigma = \{a, b\}$, $\Gamma = \Sigma \cup (\Sigma \times \{\_\}) \cup \{\sqcup\}$.

האלגוריתם:

(1) נסרוק ימינה עד למציאת רווח, ונחליף אותו ב-#.
(2) נסרוק שמאלה עד רווח או עד אות מסומנת, ונצעד צעד נוסף ימינה.
(3) מתואר בתרשים הבא:




(4) עבור למשל המצב $q_a$, סריקה ימינה עד רווח ונחליף אותו ב-$a$. נחזור לסריקה שמאלה:




<u>זמן ריצה</u>: $O(n^2)$. בכל פעם שמעתיקים אות הולכים עד צד שמאל ואז עד ימין אז עבור כל אות $O(n)$.

## כפל מספרים בינאריים

<u>קלט</u>: $x\#y$ <u>פלט</u>: $x \cdot y\#$ (כאשר $\cdot$ סימן הכפל בין מספרים בינאריים).

<u>אלגוריתם</u>:

(1) הכפלת $x$ פעם אחת, נקבל: $x\#x\#y$.
(2) הכפלת $x$ פעם נוספת, נקבל: $\boldsymbol{x}\#x\#x\#y$ (לאורך הפתרון, נשאיר את $\boldsymbol{x}$ ההתחלתי כדי שנזכור אותו).
(3) סכימת האיברים באמצע: $\boldsymbol{x}\#x + x\#y$.
(4) נוריד מ-$y$ אחד, נקבל $\boldsymbol{x}\#2x\#y - 1$.
(5) נחזור לשלב (2).

# שבוע 5 – הרצאה

## קריאה למ"ט כפרוצדורה/פונקציה

דוגמה להמחשה – ספירת צעדי חישוב

משימה: בהינתן מ"ט $M$ רוצים לבנות מ"ט $M'$ עם התכונה הבאה: אם $M$ עוצרת על $w$, $M'$ תעצור על $w$ ויתקיים $M'(w) = M(w)\#t$ כאשר $t$ הוא הייצוג הבינארי של מספר צעדי החישוב ש-$M$ מבצעת על $w$. ($M$ עושה סימולציה של $M$ אבל גם סופרת כמה צעדים). נניח ש-# לא חלק מא"ב העבודה של $M$.

הרעיון: נעשה צעד של $M$, נוסיף 1 למונה, נחזור ל-$M$ ושוב למונה וכן הלאה.

נתחיל עם מכונת טיורינג $S$:

(1) סורקת ימינה עד שמוצאת #.

(2) אם אין ספרה מימין ל-#, $S$ תכתוב 0.

(3) נוסיף 1 למונה.

(4) נסיים במצב $S_f$.

מכונת טיורינג $M'$:

(1) תוסיף # מימין לקלט, ותחזור לתחילתו.

(2) נעבור למצב ההתחלתי של $M$.

(3) נסמלץ צעד חישוב של $M$.

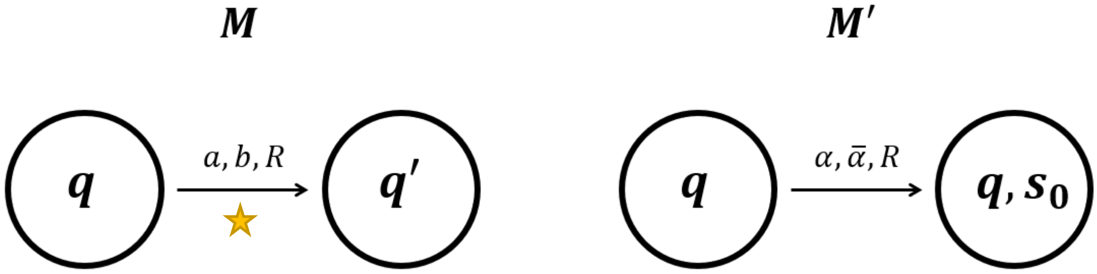
(4) "נקרא" ל-$S$ כדי להוסיף 1 למונה.

(5) נחזור חזרה למקום המקורי (בו היה נמצא הראש לפני שקראנו ל-$S$), ולמצב המקורי.

(6) נחזור לשלב (3).

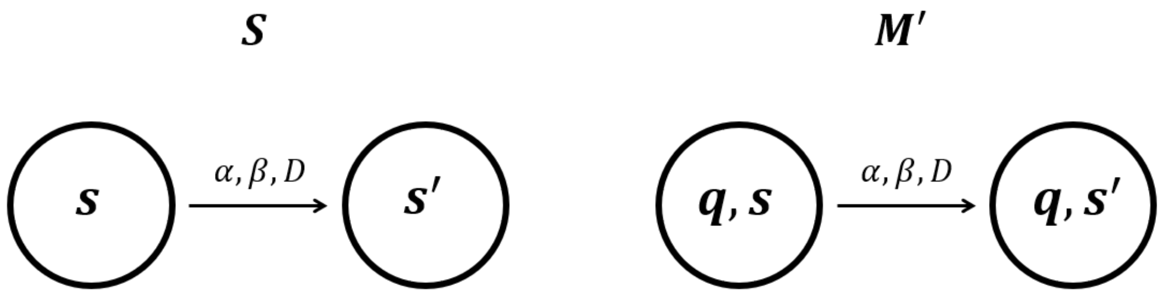
הערה: את העלאת המונה אפשר לעשות לפני $M$ או אחרי $M$, לבחירתנו.

**"קריאה" ל-$S$:** לצורך ההסבר, נניח שאנחנו אחרי שלב 3 ואנחנו במצב $q$ של המכונה $M$, ורוצים לעבור לשלב 4. אפשר לסמן באיזה מצב היינו בעזרת $\alpha^q$, מרחיב מאוד את הא"ב. כאן נשתמש ב-$\bar{\alpha}$.



איך נזכור את המצב ממנו באנו? לא נוסיף מ-$q$ חץ ל-$s_0$, אלא למצב שנגדיר $(q, s_0)$. נוסיף מצב $(q, s)$ לכל $q \in Q_M$ ו-$s \in Q_S$. ב-$M'$ יהיו את כל המצבים ב-$M$ אבל גם את כל הזוגות שהגענו, וכך מתבצעת ה"קריאה" ל-$S$.

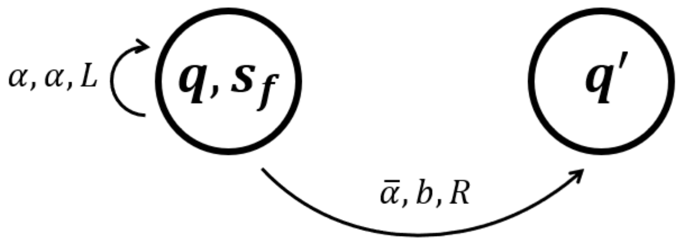
**הביצוע של $S$:**




בצורה זו $S$ ממשיך לרוץ (ילך ימינה, יעלה את המונה באיזשהו שלב יסיים כאשר נגיע ל-$(q, s_f)$.

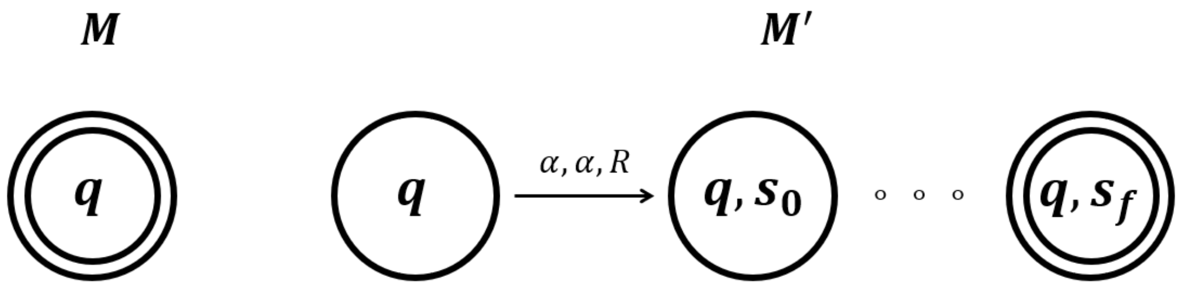
**סיום $S$:**

לכל $\alpha$ לא מסומנת:




כשנגיע לאות מסומנת, לא נחזור למצב המקורי (מסרבל). במקום זאת, נסמלץ את צעד החישוב הבא (מסומן בעמוד הקודם ב-★).

לכל חץ $q' \xrightarrow{\alpha,b,D} q$ יהיה חץ מתאים $q' \xrightarrow{\bar{\alpha},b,D} (q, s_f)$. אילו חצים יוצאים מ-$(q, s_f)$? כל החיצים שיצאו מ-$q$ (עבור אותיות מסומנות), ולולאות (עבור אותיות לא מסומנות).



עבור מצב $q$ סופי, לא נסמן את $\alpha$ ובשנגיע ל-$q, s_f$ נסיים.

נקודה למחשבה: כשמסמלצים את $M$ אנחנו יכולים להפריע לסימולציה שלו, כי הוא למשל יצפה לפגוש ␣ ויתקל ב-# ששמנו כחלק מיצירת המונה, ועלולות להיות הפרעות נוספות. כדי לטפל בהן, נקרא לפרוצדורה מתאימה שתטפל בהפרעה.

## מכונת טיורינג בקלט למכונת טיורינג

### קידוד מכונת טיורינג בקלט

נרצה למצוא דרך להכניס מכונת טיורינג בקלט למכונת טיורינג.

בהינתן מ"ט $M = \langle \Sigma, \Gamma, ␣, Q, q_0, F, \delta \rangle$ לא ניקח את $M$ עצמה אלא מחרוזת שמתארת אותה. נתאר את $M$ במחרוזת מעל $\{0,1,\#,|\}$. בה"כ נניח $\Sigma \subseteq \Gamma \subseteq \mathbb{N}$. כל אות ב-$M$ ניתנת לייצוג כמספר טבעי. כמו כן, בה"כ נניח $Q \subseteq \mathbb{N}$ וגם $\Gamma \cap Q = \emptyset$ (מטעמי נוחות).

נבטא כל אספקט של $M$ בצורה הבאה (תת אינדקס $(x)_b$ משמעותו הייצוג הבינארי של $x$):

$$\Sigma = \{\alpha_1, \dots, \alpha_k\} \to \langle \Sigma \rangle = (\alpha_1)_b\# \dots \#(\alpha_k)_b\#$$

$$\Gamma = \{\alpha_1, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n\} \to \langle \Sigma \rangle = (\alpha_1)_b\# \dots \#(\alpha_k)_b\#(\alpha_{k+1})_b\# \dots \#(\alpha_n)_b$$

$$␣ = \alpha_i \to (\alpha_i)_b$$

$$Q = \{q_1, \dots, q_m\} \to \langle Q \rangle = (q_1)_b\# \dots \#(q_m)_b$$

ואת $q_0, F$ באופן דומה.

עבור פונקציית המעברים אם מתקיים $\delta(q,\alpha) = (q', \beta, D)$ אז נגיד ש-$(q,\alpha) \xrightarrow{\delta} (q', \beta, D)$ הוא כלל מעבר (ייצוג גרפי של המעבר). נייצג את $\delta$ ע"י שרשור כלל הייצוגים של כללי המעבר שהיא מגדירה. כל מעבר ייוצג:

$$\#\#(q)_b\#(\alpha)_b\#(q')_b\#(\beta)_b\#(D)_b\#\#$$

לבסוף, על מנת לייצג את כל $\langle M \rangle$: $\langle M \rangle = \langle \Sigma \rangle | \langle \Gamma \rangle | \langle ␣ \rangle | \dots | \langle \delta \rangle |$

### קידוד הקלט למכונת טיורינג

$$\langle w \rangle = (w_1)_b\#(w_2)_b\# \dots$$

### קידוד קונפיגורציה

$$\langle C \rangle = (\alpha_1)_b\#(\alpha_2)_b\# \dots \#(\alpha_{i-1})_b\# \dots \#(\alpha_m)_b$$

לבסוף, כדי להעביר את המכונה $M$ והמילה  והמילה $w$ בקלט למכונת טיורינג, נבצע:

$$\langle M, w\rangle = \langle M\rangle\langle w\rangle$$

**<u>הגדרה – מכונת טיורינג אוניברסלית:</u>** מ"ט $\boldsymbol{U}$ תיקרא **אוניברסלית** אם בהינתן $\langle \boldsymbol{M}, \boldsymbol{w}\rangle$:

(1) אם $M$ לא עוצרת בריצתה על $w$ אז $U$ גם לא תעצור.

(2) אם $M$ עוצרת בריצתה על $w$ אז $U$ תעצור, ויתקיים: $U(\langle M, w\rangle) = \langle M(w)\rangle$.

(3) אם המצבים הסופיים של $M$ הם $q_{rej} = 0, q_{acc} = 1$ אז $U$ יסיים באותו מצב כמו $M$.

<u>טענה</u>

קיימת מ"ט אוניברסלית $U$.

<u>רעיון הוכחה</u>: $U$ תפעל באופן הבא:

(0) קלט $\langle M, w\rangle$

(1) תכתוב על הסרט

(2) $\langle M\rangle\langle C\rangle \to \langle M\rangle\langle C^+\rangle$ בלולאה עד קונפיגורציה סופית, אם קיימת.

(3) אם נגיע לקונפיגורציה סופית:

(א) נמחק את $\langle M\rangle$

(ב) נמחק את $(q)_b$

(ג) אם $q \in \{0,1\}$ נסיים במצב $q$ (נובע מתכונה 3 בהגדרה). אחרת נסיים ב-$q_f$ כלשהו.

<u>הערה</u>: יש כאן הרבה בחירות והחלטות, כמעט אין שום דרך לעשות את זה בצורה שמגדילה מאוד את זמן הריצה. אם $M$ רץ ב-$O(t)$ פעולות אז נגיע לזמן שהוא פולינומיאלי ב-$t$ ולא אקספוננציאלי. ניתן להגיע אפילו ל-$O(t\log t)$ עם הרבה אופטימיזציות.

## בעיית העצירה

<u>האם $M$ עוצרת על המילה $w$?</u>

**<u>הגדרה – $\boldsymbol{HALT}$:</u>** שפה $\boldsymbol{HALT}$ (בספרות לפעמים $\boldsymbol{HALT_{TM}}$) תוגדר להיות:

$$HALT = \{\langle M, w\rangle \mid M \text{ stops on } w\}$$

מכילה את כל המחרוזות שהן תיאור של מכונה עם $w$, אבל רק כאלו ש-$M$ עוצרת על $w$.

**<u>הגדרה – מכונת החלטה:</u>** מ"ט $\boldsymbol{M}$ נקראת מכונת החלטה אם קבוצת המצבים הסופיים שלה מורכבת מהמצבים $\boldsymbol{F} = \{\boldsymbol{q_{acc}}, \boldsymbol{q_{rej}}\}$.

השפה של מכונה כזו $\boldsymbol{L(M) = \{w \mid M \text{ stops at } q_{acc} \text{ when running on } w\}}$.

**הגדרה – M מקבלת את המילה $w$:** אם $\boldsymbol{M}$ עוצרת במצב $\boldsymbol{q_{acc}}$ בריצתה על $\boldsymbol{w}$.

**הגדרה – M דוחה את המילה $w$:** אם $\boldsymbol{M}$ עוצרת במצב $\boldsymbol{q_{rej}}$ בריצתה על $\boldsymbol{w}$.

**הגדרה – M מזהה את השפה L:** נאמר שמ"ט $\boldsymbol{M}$ מזהה את השפה $\boldsymbol{L}$ כאשר $\boldsymbol{L(M) = L}$.

**הגדרה – M מכריעה את השפה L:** נאמר שמ"ט $\boldsymbol{M}$ מכריעה את השפה $\boldsymbol{L}$ כאשר $\boldsymbol{L(M) = L}$ וגם עוצרת על כל קלט.

**הגדרה - $RE$:** $\boldsymbol{RE = \{L \mid Exists\ a\ TM\ that\ recognizes\ L\}}$.

טענה

קיימת מ"ט שמזהה את $HALT$, כלומר $HALT \in RE$.

הוכחה: בעזרת קיום מ"ט אוניברסלית. נהפוך כל מצב סופי של המכונה האוניברסלית למצב סופי מקבל.

(מכונה אוניברסלית מקבלת $\langle M, w\rangle$, אם $M$ לא עוצרת על $w$ האוניברסלית לא תעצור, ואם היא כן עוצרת אז גם האוניברסלית תעצור).

**הגדרה - $R$:** $\boldsymbol{R = \{L \mid Exists\ a\ TM\ that\ decides\ L\}}$.

טענה

אין מ"ט שמכריעה את $HALT$, כלומר $HALT \notin R$.

הוכחה: נניח בשלילה שקיימת מ"ט $D$ שמכריעה את $HALT$.

נבנה ממנה מ"ט $E$ שעבור קלט $\langle M\rangle$ פועלת כדלהלן:

(1) $E$ תכפיל את הקלט ונקבל: $\langle M, \langle M\rangle\rangle$.

(2) $E$ תפעיל את $D$ כפרוצדורה.

(3) אם $D$ מחזירה "כן", $E$ תכנס ללולאה אינסופית. אם $D$ מחזירה "לא" אז $E$ תקבל (בפרט, תעצור).

כעת, נריץ את $E$ על הקלט $\langle E\rangle$. האם $E$ עוצרת על קלט זה?

נניח **שכן**. אז $D$ תקבע ש-$E$ תעצור, ואז מהאופן בו הגדרנו את $E$ היא תכנס ללולאה אינסופית – בסתירה.

נניח **שלא**. באופן דומה, $D$ תקבע ש-$E$ לא עוצרת, ואז $E$ דווקא תעצור. שוב, סתירה.

שני המקרים הובילו לסתירה. מצב זה נובע מהנחת השלילה שקיימת מ"ט $D$ שמכריעה את $HALT$.

**הגדרה – $HALT_\epsilon$:** שפה $HALT_\epsilon$ תוגדר להיות: $HALT_\epsilon = \{\langle M \rangle \mid M \text{ stops on } \epsilon\}$

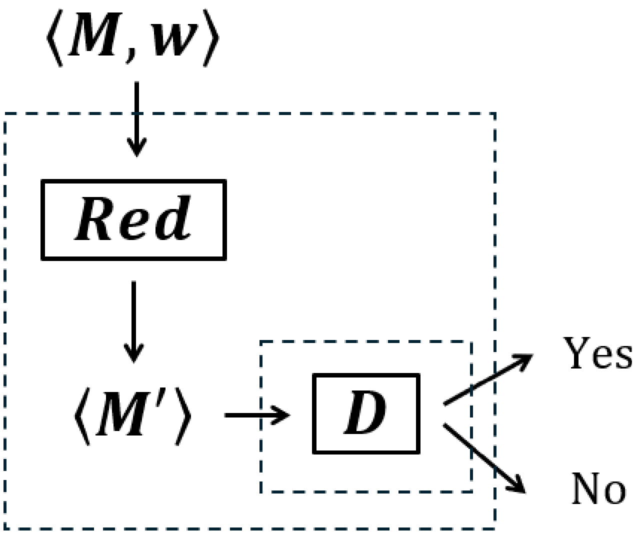
טענה: $HALT_\epsilon \notin R$

הוכחה: ניתן לבנות מ"ט $Red$ (מלשון רדוקציה) שפועלת באופן הבא על קלט $\langle M, w \rangle$:

$Red$ תייצר תיאור של מ"ט $M'$ אשר בהינתן קלט ריק מדפיסה את $w$ ואז רצה עליו כמו המכונה $M$:

$$M\langle M, w\rangle \xrightarrow{Red} \langle M' \rangle$$

נניח בשלילה ש-$D$ מ"ט המכריעה את $HALT_\epsilon$. מכונה זו גם תכריע את השפה $HALT$ כי בעזרתה נוכל עבור כל מ"ט $M$ ומילה $w$ לדעת האם המכונה תעצור. זה כמובן לא מתאפשר (הוכחנו לעיל $HALT \notin R$) ולכן הגענו לסתירה.



# שבוע 6 – הרצאה

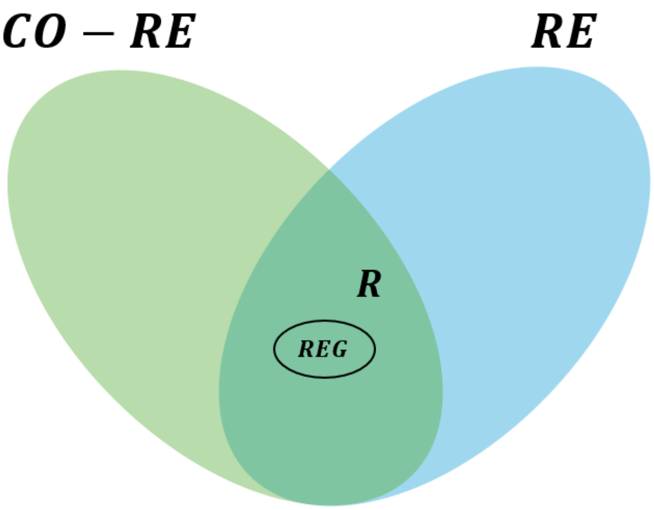
## יחסים בין השפות R, RE

בהרצאה הקודמת הגדרנו את R ואת RE. היום לאורך השיעור נפתח את היחסים בין השפות ונעמיק את היחס.

**הגדרה - $CO-RE$:** נגדיר: $\boldsymbol{CO-RE=\{L\mid \bar{L}\in RE\}}$ ואפשר להגדיר גם באופן שלא תלוי ב-$\bar{L}$ כך:

$$CO-RE=\{L\mid exists\ TM\ M\ that\ rejects\ input\ x\Leftrightarrow x\notin L\}$$

במהלך ההרצאה נרחיב את הדיאגרמה הבאה:




טענה- $R\subseteq CO-RE$.

(1) האם קיים משהו ב-$RE\cap CO-RE$ חוץ מ-$R$? לא. נוכיח כי $RE\cap CO-RE=R$.

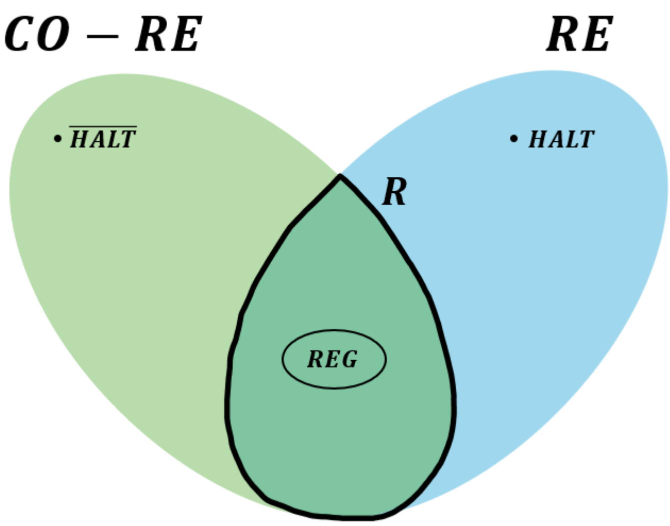
צד אחד ברור, $R\subseteq RE\cap CO-RE$.

נראה צד שני: $R\supseteq RE\cap CO-RE$.

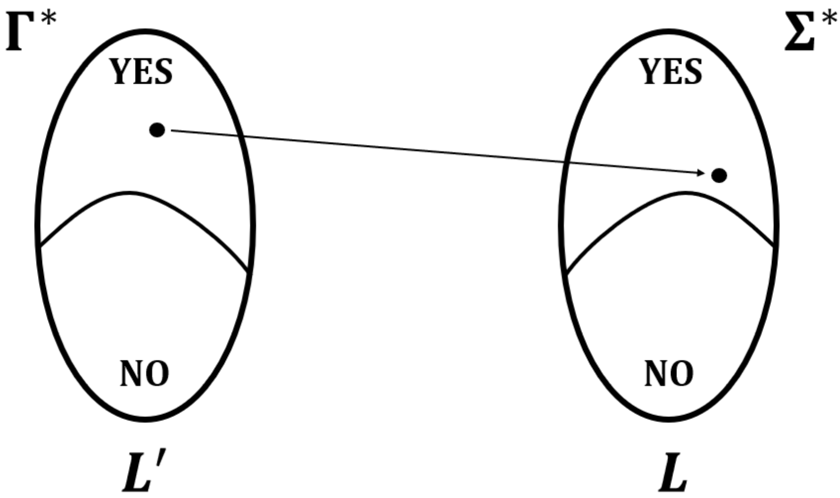
תהי $L\in RE\cap CO-RE$, אז יש מ"ט $M_L$ ו-$M_{\bar{L}}$ שמזהות את $L$ ואת $\bar{L}$ בהתאמה. ניתן לבנות מ"ט $M_D$ שמכריעה את $L$ ע"י שבהינתן קלט $x$ המכונה $M_D$ תפעיל לכל $i\in\mathbb{N}$ את $M_L$ ואת $M_{\bar{L}}$ על $x$ למשך $i$ צעדים כל אחת. אם $M_L$ תקבל/תדחה, אז $M_D$ תקבל/תדחה בהתאמה. אם $M_{\bar{L}}$ תקבל/תדחה, אז $M_D$ תדחה/תקבל בהתאמה. מובטח שלפחות אחד מהדברים יקרה לבסוף, ולכן ל-$M_D$ תהיה תשובה.

(2) האם יש שפה ב-$CO-RE$ אבל לא ב-R? כן, $\overline{HALT}$. מהגדרתה, אם הייתה ב-$RE$ אז היתה גם ב-$R$ ולכן היה אפשר להכריע אותה, ואנחנו יודעים שאי אפשר, לכן $\overline{HALT}\in CO-RE\setminus RE$.

נעדכן את הדיאגרמה לאור הטענה:




## רדוקציה




נתעניין במ"ט $Red$, שלא פותרת לא את $L$ ולא את $L'$, אלא מתרגמת שאלה מעל $L'$ לשאלה מעל $L$, כשכל מה שידוע לה זה שלשתיהן יש אותה תשובה לשאלה.

**הגדרה – מכונת רדוקציה:** $\Sigma^*$ $\boldsymbol{L} \subseteq$, $\boldsymbol{L'} \subseteq \boldsymbol{\Gamma}^*$. מכונת רדוקציה מ-$\boldsymbol{L}$ ל-$\boldsymbol{L'}$ היא מ"ט $\boldsymbol{Red}$ שעוצרת על כל קלט $\boldsymbol{x} \in \boldsymbol{\Gamma}^*$. ומחזירה מילה מעל $\boldsymbol{\Sigma}$ כך שמתקיים $\boldsymbol{Red}(\boldsymbol{x}) \in \boldsymbol{L'} \Leftrightarrow \boldsymbol{x} \in \boldsymbol{L}$.

אם קיימת מ"ט כנ"ל, נגיד שקיימת רדוקציה מ-$L'$ ל-$L$. לפונקציה $Red(x)$ $x \mapsto$ נקרא פונקציית רדוקציה מ-$L'$ ל-$L$, ונסמן $L' \leq_m L$.

למה מסמנים $\leq$? $L$ היא לפחות קשה מבחינת פתרון כמו $L'$, לכן אם היינו פותרים את $L$, נדע לפתור גם את $L'$.

משפט הרדוקציה- אם $L' \leq_m L$ אז:

(1) אם $L \in R$ אז $L' \in R$.
(2) אם $L \in RE$ אז $L' \in RE$.
(3) אם $L \in CO-RE$ אז $L' \in CO-RE$.

טענה- $HALT \leq_m A_{TM}$ כאשר $A_{TM} = \{\langle M, w\rangle \mid M\ accepts\ w\}$.

רעיון הוכחה: את $M'$ נשנה כך שרק כאשר $M$ המקורית דוחה, $M'$ תקבל.

הוכחה: הרדוקציה תחזיר בהינתן קלט $\langle M, w\rangle$ קידוד של $\langle M', w\rangle$ כאשר $M'$ היא מ"ט שפועלת כמו $M$ למעט ש-$M'$ מקבלת גם כש-$M$ דוחה.

נכונות: הרדוקציה המתוארת אכן ניתנת למימוש על ידי מ"ט שעוצרת תמיד.

$$\langle M', w\rangle \in A_{TM} \Leftrightarrow M'\ accepts\ w \Leftrightarrow M\ stops\ w \Leftrightarrow \langle M, w\rangle \in HALT$$

טענה- $A_{TM} \leq_m HALT$.

רעיון: המקרה הבעייתי הוא אם $M$ דוחה את $w$. $\langle M, w\rangle \notin A_{TM}$, אבל $\langle M, w\rangle \in HALT$ וזה המקרה שבו צריך לטפל.

הוכחה: בהינתן $\langle M, w\rangle$ הרדוקציה תחזיר $\langle M', w\rangle$ כאשר $M'$ היא מ"ט שפועלת כמו $M$ למעט המקרה בו $M$ דוחה, ואז $M'$ תכנס ללולאה אינסופית.

נכונות: הרדוקציה היא חשיבה, ומתקיים:

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow M\ accepts\ w \Leftrightarrow M'stops\ (and\ accepts)\ w \Leftrightarrow \langle M', w\rangle \in HALT$$

**הגדרות – $\mathcal{C}$-קשה, $\mathcal{C}$-שלמה:** אם $\mathcal{C}$ מחלקת שפות ו-$L$ כך שמתקיים $L' \leq_m L$ לכל $L' \in \mathcal{C}$ אז נאמר ש-$L$ היא $\mathcal{C}$-קשה. אם בנוסף מתקיים ש-$L \in \mathcal{C}$, נאמר ש-$L$ היא $\mathcal{C}$-שלמה.

טענה- $A_{TM}$ היא $RE$-שלמה.

הוכחה: $A_{TM} \in RE$ . נשאר להראות שאם $L \in RE$ אז $L \leq_m A_{TM}$. עבור $L$ כזו, יש מ"ט $M_L$ שמזהה אותה. בהינתן קלט $x$ עבור $L$, הרדוקציה תחזיר $\langle M_L, x\rangle$ וקיבלנו $\langle M_L, x\rangle \in A_{TM} \Leftrightarrow x \in L$.

טענה- השפה $\overline{HALT}$ היא $CO-RE$-שלמה.

הוכחה: תהא $L \in CO-RE$. אזי $\bar{L} \in RE$, לכן $\bar{L} \leq_m HALT$, ועם אותה רדוקציה נוכל לקבל גם $L \leq_m \overline{HALT}$.

**הגדרה - $Non-HALT$:** נגדיר $E$ $\overline{HALT}=\{\langle M,w\rangle \mid M\ doesn't\ stop\ on\ w\}\cup E$ כאשר $E$ קבוצת קלטים לא חוקיים "$Error$". את החלק השני, נגדיר כ-$Non-HALT$.

טענה- $Non-HALT$ היא שלמה ב-$CO-RE$.

הוכחה: נראה כי:

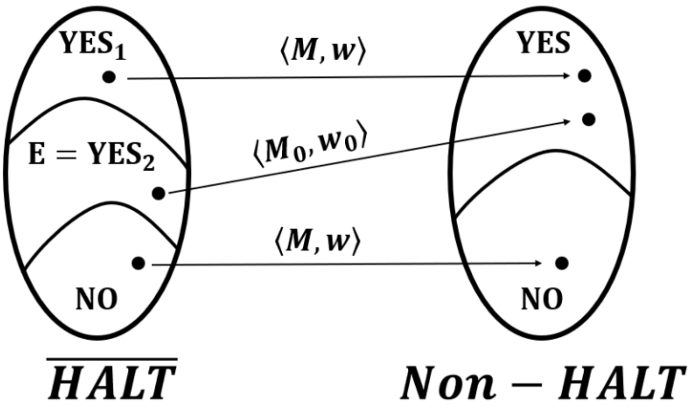
(1) $Non-HALT\in CO-RE$. נוכיח זאת באמצעות כך שנראה $\overline{Non-HALT}\in RE$.

$$\overline{Non-HALT}=HALT\cup E$$

הראינו $HALT\in RE$, ומתקיים $E\in R$ (כי קלטים לא חוקיים ניתנים להכרעה האם מתקבלים או לא), ולכן גם $E\in RE$.

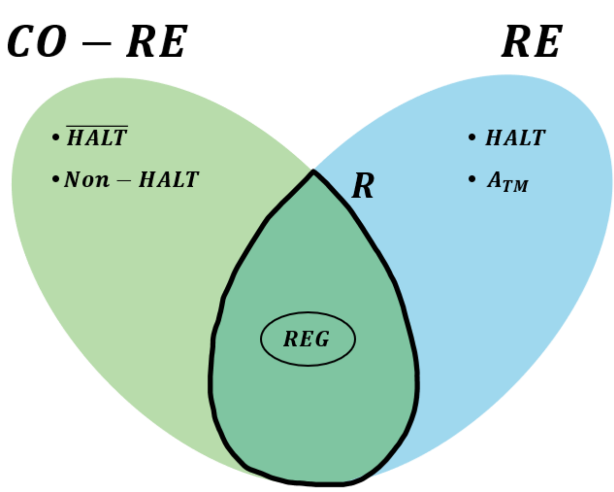
מסגירות $RE$ לאיחוד נקבל $\overline{Non-HALT}\in RE$ ולכן $Non-HALT\in CO-RE$.

(2) נסיים ברדוקציה מ-$\overline{HALT}$ ל-$Non-HALT$.




במיפוי מ-YES ל-YES או מ-NO ל-NO, נשמור על הערך כפי שהוא. ב-$YES$ נחזיר $\langle M,w\rangle$ כך ש-$M$ לא עוצר על $w$, וב-$NO$ נחזיר $\langle M,w\rangle$ כך ש-$M$ עוצר על $w$. הבעיה נוצרת במעבר מקלט $x$ כלשהו שמקודד לערך לא מוגדר. מאחר ש-$x$ כן ב-$\overline{HALT}$ אנחנו צריכים להחזיר משהו שישמור על ערך האמת, כלומר משהו שכן ב-$Non-HALT$. במקרה זה נחזיר $\langle M_0,w_0\rangle$ שזו קידוד למכונה וקלט ידועים מראש כך שהמכונה תמיד לא עוצרת על הקלט שבחרנו.

נעדכן את הדיאגרמה:



מה לגבי שפות "קשות יותר"? מכל שפה ב-$RE$ וב-$CO-RE$? (כלומר שיש רדוקציה אליהן מכל שפה ב-$RE$ וב-$CO-RE$). דוגמאות:

$$\text{Y-HALT} = \{\langle M,w\rangle, Y \mid M\ halts\ on\ w\}$$

$$\text{N-HALT} = \{\langle M,w\rangle, N \mid M\ doesn't\ halt\ on\ w\}$$

$$\text{NY-HALT} = Y-HALT \cup N-HALT$$

נשים לב שכדי לדעת אם מילה נמצאת ב-NY-HALT אנחנו נקבל $\langle M,w\rangle$ ונצטרך לדעת אם להוסיף אחרי הקידוד את $N$ או את $Y$ – ולשם כך צריך לדעת אם $M$ עוצרת על $w$.

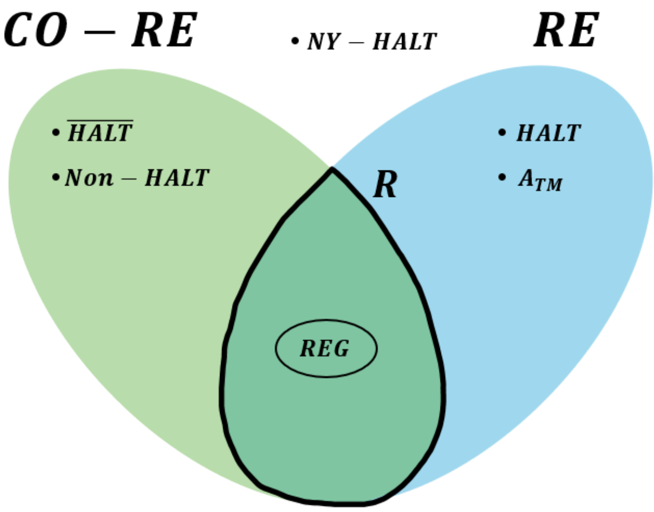
<u>טענה</u>- NY-HALT היא גם $RE$ וגם $CO-RE$ קשה.

<u>הוכחה (RE)</u> : נראה רדוקציה מ-$HALT$. בהינתן $\langle M,w\rangle$ הרדוקציה תחזיר $\langle M,w\rangle, Y$.

<u>הוכחה (CO-RE)</u>: רדוקציה מ-Non-HALT. בהינתן $\langle M,w\rangle$ הרדוקציה תחזיר $\langle M,w\rangle, N$.

<u>טענה</u>- NY-HALT $\notin RE$.

<u>הוכחה</u>: אחרת, ממשפט הרדוקציה אנחנו נקבל Non-HALT $\in RE$.. נתגלגל משם עד שנגיע לסתירה לדברים שראינו.




<u>התזה של טיורינג וצ'רץ'</u>- כל מכונת חישוב (ביקום שלנו) ניתנת לסימלוץ על ידי מ"ט.

**<u>הגדרה – DECIDABLE</u>**: נגדיר $\text{DECIDABLE} = \{\boldsymbol{claims\ we\ can\ prove\ or\ contradict}\}$.

טיורינג טען $DECIDABLE \notin R$.

<u>הוכחה</u>: נניח בשלילה כי $T$ מ"ט שמכריעה את $DECIDABLE$. נראה שזה יגרור $HALT \in R$:

נבנה מ"ט $S$ שבהינתן $\langle M,w\rangle$, $S$ תכתוב את הטענה "$M$ לא עוצרת על $w$", ואז תפעיל את $T$ על הטענה:

(1) אם $T$ ענתה "לא" (כלומר אי אפשר להוכיח או להפריך את הטענה) אז $S$ תחזיר "לא".

(2) אם $T$ ענתה "כן" (כלומר אפשר להוכיח או להפריך את הטענה) אז $S$ תרוץ על כל ההוכחות האפשריות.

ללא תלות בתשובה של $T$ נוכל להכריע האם $M$ תעצור על $S$, כלומר מצאנו דרך להכריע את $HALT$ ולכן $HALT \in R$, בסתירה.

# שבוע 7 – הרצאה

## סיבוכיות

מה שניתן לחשב במשאבי חישוב מוגבלים. נדון רק בבעיות חישוביות כריעות, ונתעניין בעיקר כמה זמן/מקום על הסרט נדרש כדי להכריע.

בהרצאה זו מופיעות הגדרות רבות על זמן/מקום ריצה. בכל מקום שבו ניתן הגדרה עבור סיבוכיות הזמן, אפשר גם לתת הגדרה דומה עבור סיבוכיות מקום.

**הגדרה – זמן ריצה:** $\boldsymbol{M}$ מ"ט, $\boldsymbol{w}$ קלט, זמן הריצה של $\boldsymbol{M}$ על $\boldsymbol{w}$ הוא אורך הריצה המלאה של $\boldsymbol{M}$ על $\boldsymbol{w}$, פחות 1 (עבור הקונפיגורציה ההתחלתית).

**הגדרה – מקום ריצה:** מספר התאים שהראש מגיע אליהם בזמן הריצה.

**הגדרה – זמן הריצה של $\boldsymbol{M}$:** נתייחס למקרה הגרוע ביותר. אם $\boldsymbol{M}$ מכונת טיורינג, זמן הריצה שלה הוא פונקציה $\boldsymbol{T}: \mathbb{N} \to \mathbb{N}$ המוגדרת על ידי:

$$T(n) = \max_{|w|\le n}\{\text{running time of M on w}\}$$

**הגדרה – מקום הריצה של $\boldsymbol{M}$:** נתייחס למקרה הגרוע ביותר. אם $\boldsymbol{M}$ מכונת טיורינג, מקום הריצה שלה הוא פונקציה $\boldsymbol{S}: \mathbb{N} \to \mathbb{N}$ המוגדרת על ידי:

$$S(n) = \max_{|w|\le n}\{\text{running space of M on w}\}$$

**הגדרה – חסימת זמן ומקום הריצה:** נגיד שמ"ט $\boldsymbol{M}$ רצה בזמן $\boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))$ אם $\boldsymbol{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))$. אפשר להגדיר גם עבור $\boldsymbol{S}(\boldsymbol{n})$ וכן עבור החסמים $\boldsymbol{o}, \boldsymbol{\Omega}, \boldsymbol{\omega}, \boldsymbol{\Theta}$. נניח ש-$\boldsymbol{f}$ היא מונוטונית תמיד.

**הגדרה - $\text{TIME}\left(\boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))\right)$:** תהא $\boldsymbol{f}: \mathbb{N} \to \mathbb{N}$ מונוטונית עולה. נגדיר:

$$\text{TIME}\left(O(f(n))\right) = \{L \mid \exists \text{ TM that decides } L, \text{and runs in } \mathrm{O}(\mathrm{f}(\mathrm{n})) \text{ time}\}$$

**הגדרה - $\text{SPACE}\left(\boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))\right)$:** תהא $\boldsymbol{f}: \mathbb{N} \to \mathbb{N}$ מונוטונית עולה. נגדיר:

$$\text{SPACE}\left(O(f(n))\right) = \{L \mid \exists \text{ TM that decides } L, \text{and runs in } \mathrm{O}(\mathrm{f}(\mathrm{n})) \text{ space}\}$$

נשים לב שהגדרות אלו מוגדרות רק עבור בעיות הכרעה, כי מניחים שקיימת $TM$ מכריעה.

### דוגמה 1 - INPUT-LENGTH

מתבצע ב-$O(n \log n)$ זמן. ראינו איך לעשות זאת בתרגילי הבית – אפשר למקם מונה משמאל וכל פעם להעלות אותו ב-1 (בזמן $\log n$) ואז להעתיק אותו ימינה ב-1 (בזמן $\log n$). יש $n$ איטרציות כאלו ולכן הזמן הכולל הוא $O(n \log n)$. אי אפשר לבצע בפחות זמן, אך לא קל להראות את זה.

### דוגמה 2 - $L \in$ REG

זמן $O(n)$ וגם מקום $O(n)$.

### דוגמה 3 - PALINDROM

שפת המילים שהן והיפוכן זהות. לזהות ייקח $\Theta(n^2)$ זמן, כי צריך $n$ טיולים אחורה וקדימה בקלט, במקרה הגרוע טיול אורך $n$. $O(n)$ זמן. במ"ט דו-ראשית נוכל להוריד לזמן $\Theta(n)$.

**הגדרה - POLYNOMIAL TIME:** נגדיר P (לפעמים גם PTIME) להיות:

$$\mathrm{P} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}\left(O(n^k)\right)$$

**הגדרה - POLYNOMIAL SPACE:** נגדיר PSPACE להיות:

$$\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{SPACE}\left(O(n^k)\right)$$

הערה: ההגדרה של P כמעט ואינה תלוית מודל. לפי תזת צ'רץ'-טיורינג המורחבת כל מחשב שניתן לבנות ביקום הפיזי שלנו, ורץ בזמן $O\left(f(n)\right)$ ניתן לסמלץ במ"ט רגילה בזמן $O(f(n)^k)$ עבור $k$ קבוע כלשהו. התזה גוררת ש-P אינו תלוי במודל החישובי.

**הגדרה - E:** נגדיר E להיות:

$$\mathrm{E} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}\left(O((2^k)^n)\right)$$

**הגדרה - EXP:** נגדיר EXP להיות:

$$\mathrm{EXP} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}\left(O\left(2^{n^k}\right)\right)$$

### דוגמה 1 - CONNECTED

נגדיר: $\mathrm{CONNECTED} = \{\langle G\rangle \mid G \textit{ is a connected graph}\}$. בהינתן גרף, האם ניתן בזמן פולינומי להכריע האם הוא קשיר? כן. ראינו בקורסים קודמים באמצעות $BFS/DFS$. לכן, $\mathrm{CONNECTED} \in \mathrm{P}$.

### דוגמה 2 - MIN-SPANNING-TREE

האם בעיה זו ב-P? ללא שום שינוי, לא מדובר בבעיית הכרעה ולכן לא נוכל לדבר על P. נגדיר:

$$\text{MIN-SPANNING-TREE} = \{\langle G\rangle, \langle w\rangle, (k)_b \mid \text{There is a spanning tree in } G \text{ in weight} \leq \text{k}\}$$

בעיה זו כן נמצאת ב-P.

דוגמה 3 - SIMP-PATH

נגדיר $\text{SIMP-PATH} = \{\langle G \rangle, k \mid \exists \text{ simple path of length } k \text{ in graph } G\}$. נטען כי בעיה זו היא ב-E.

הסיבה לכך היא שנעבור על כל $\binom{m}{k}$ $k$-יות של קשתות, ונבדוק האם הן מסלול פשוט.

זמן הריצה: $\binom{m}{k} \le \binom{n}{k} \overset{\text{binom}}{\le} 2^n$

## מכונת טיורינג לא דטרמיניסטית

תזכורת: במכונת טיורינג דטרמיניסטית ראינו לגבי פונקציית המעברים: $\delta(q,a) \in Q \times \Gamma \times \{R,L\}$.

במכונת טיורינג לא דטרמיניסטית, $\delta$ תחזיר תת-קבוצה של $Q \times \Gamma \times \{R,L\}$ עבור כל הריצות האפשריות.

הגדרות נוספות עבור מכונת טיורינג לא דטרמיניסטית (מ"ט ל"ד):

**הגדרה – ריצה חלקית:** סדרה של קונפיגורציות שבה הראשונה היא קונפיגורציה התחלתית (ניזכר שיתכנו כמה מצבים התחלתיים). כל קונפיגורציה שתגיע אחרי היא עוקבת של הקודמת (אחת מבין החוקיות).

**הגדרה – ריצה מלאה:** ריצה חלקית אינסופית, או שמסתיימת בקונפיגורציה סופית.

הערה: אם הגענו למצב ממנו אין חצים יוצאים אבל זה לא מצב סופי – הריצה תקועה, ואינה ריצה מלאה.

**הגדרה – מ"ט ל"ד מקבלת מילה $w$:** אם קיימת ריצה מקבלת של המכונה על $\boldsymbol{w}$.

**הגדרה – מ"ט ל"ד מזהה שפה $L$:** מ"ט ל"ד $\boldsymbol{M}$ מזהה את השפה $\boldsymbol{L}$ כאשר: $\boldsymbol{w} \in \boldsymbol{L} \Leftrightarrow \boldsymbol{M} \text{ accepts } \boldsymbol{w}$.

**הגדרה - NP:** נגדיר את NP להיות:

$$\text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}\left(O(n^k)\right)$$

באופן דומה להגדרות שראינו לעיל, נוכל להגדיר:

$$\text{NPSPACE} = \bigcup_{k \in \mathbb{N}} \text{NSPACE}\left(O(n^k)\right)$$

$$\text{NE} = \bigcup_{k \in \mathbb{N}} \text{NTIME}\left(O((2^k)^n)\right)$$

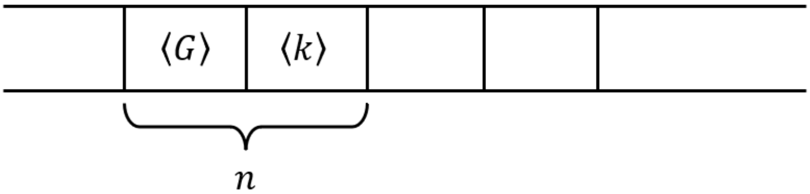
$$\text{NEXP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}\left(O\left(2^{n^k}\right)\right)$$

מה אנחנו יודעים להגיד על היחס בין P ו-NP? $\text{NP} \supseteq \text{P}$, כי כל מכונה דטרמיניסטית אפשר להסתכל עליה גם כלא דטרמיניסטית.

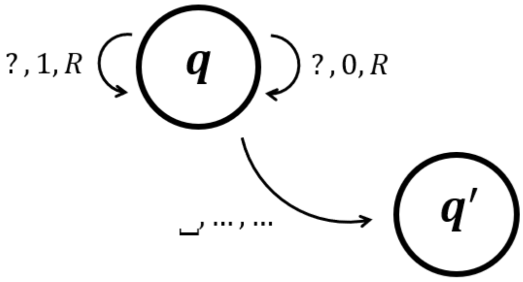
## דוגמה – בעיית HAMILTON-PATH

מוגדרת להיות: $\text{HAMILTON-PATH} = \{\langle G\rangle \mid \exists \text{ a path of length } |V(G)|-1\}$

בבעיית SIMP-PATH נניח שהקלט על הסרט נראה כך:




בהנחה שהקלט הוא בגודל $n$, אז לא ייתכן שבדי לייצג רק את הקשתות נצטרך יותר מ-$n$ מקום, לכן נניח $n$. אם נרצה לתאר קשת אחת נצטרך $\log(n)$. כדי לתאר $k$ קשתות צריך $k\log n$. המכונה תספור את אורך הקלט $n$, תחשב את $\log n$ ואז תכתוב $k\log n$ סימני שאלה, תחזור ל-"?" הראשון ותעבור למצב מנחש:




יש $2^{k\log n}$ ריצות אפשריות. נבדוק האם המחרוזת שכתבנו היא במקרה תיאור של מסלול העונה על הדרישה.

<u>זמן ריצה</u>: בכל ריצה נכתוב $k\log n$ פעמים ? ונחשב את $n$ ואת $k\log n$ ולבדוק האם זה מסלול בגרף. הכל מתבצע בזמן פולינומי. אם קיים מסלול כזה, זה יהיה באחת מהריצות ונחזיר "כן", אחרת נחזיר "לא". לכן אנחנו מסיקים $\text{SIMP-PATH} \in \text{NP}$.

## בעיית CNF-SAT (Conjunctive Normal Form)

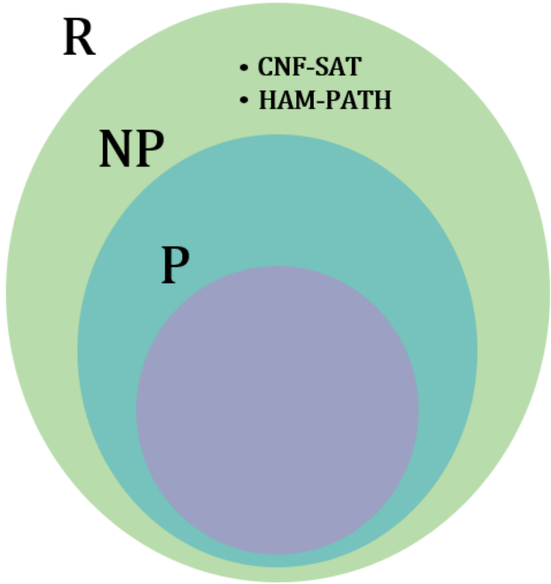
<u>קלט</u>: נוסחה בוליאנית מעל משתנים בוליאנים $x_1,\ldots,x_n$ מהצורה: $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. כל $Clause$, כלומר כל $C_i$ הוא מהצורה:

$$C_i(x_1 \vee x_2 \vee \overline{x_4} \vee x_7)$$

משתנה או שלילתו נקראים ליטרל, ולכן $C_i$ הוא "או" על ליטרלים. קלט כזה הוא בשפה אם יש השמה למשתנים שמספקת את $\varphi$.

<u>טענה</u>: $\text{CNF-SAT} \in \text{E}$. לעבור על כל השמה של המשתנים הבוליאנים ולבדוק אם יש כזו שמספקת את $\varphi$. יש $2^n$ השמות – לכל אחת נעבור על הפסוק. כמה זמן לוקח לבדוק האם השמה מספקת? לא מאוד מעניין, זמן פולינומי.

האם CNF-SAT ∈ NP? כן, כמו בדוגמת SIMP-PATH נכתוב $n$ סימני שאלה ואז נמלא 0 או 1 בכל תא, ונבדוק האם רצף זה הוא השמה מספקת.




## רדוקציה פולינומיאלית

**הגדרה – רדוקציה בזמן פולינומיאלי:** נגיד על מכונת רדוקציה מ-$L'$ ל-$L$ שהיא רדוקציה בזמן פולינומיאלי מ-$L'$ ל-$L$ כאשר היא רצה בזמן שחסום על ידי פולינום באורך הקלט שלה. אם קיימת כזו, נסמן $L' \le_p L$.

משפט – רדוקציה פולינומיאלית:

אם $L \in P$ ו-$L' \le_p L$, אז $L' \in P$.

לא נוכיח, אבל ניתן אינטואיציה:

$$\underset{n}{x} \in L' \xRightarrow[n^{k_1}]{Red} \underbrace{Red(x) \in L} \Longrightarrow \underbrace{M \circ Red(x)}_{n^{k_1 k_2}}$$

באורך לכל היותר $n^{k_1}$
כי הרדוקציה רצה בזמן $n^k$ אז לא
ייתכן שכתבה משהו ארוך יותר

עדיין בזמן פולינומיאלי, ולכן אם אפשר לפתור את $L$ בזמן פולינומי אפשר לפתור גם את $L'$ בזמן פולינומי.

**הגדרה – שפה $C$-קשה:** תהי $\boldsymbol{C}$ מחלקת שפות. $\boldsymbol{L}$ תיקרא $\boldsymbol{C}$-קשה ביחס לרדוקציה פולינומית אם $\boldsymbol{L' \leq_p L}$ לכל $\boldsymbol{L' \in C}$. היא תיקרא $\boldsymbol{C}$-שלמה אם היא $\boldsymbol{C}$-קשה וגם $\boldsymbol{L \in C}$.

אם נצליח למצוא לפתור בעיה NP-שלמה בזמן פולינומי, זה יכריע את **שאלת מיליון הדולר**, ונוכיח $\mathrm{P} = \mathrm{NP}$.

# שבוע 8 – הרצאה

<u>תזכורת</u>

$$P \subseteq NP \subseteq EXP \subseteq R$$

מ"ט דטרמיניסטית היא מקרה פרטי של מ"ט לד"ד

תרגיל בית 7

יש מכונת הכרעה (שבמקרה גם חסומה בזמן ריצה אקספוננציאלי)

<u>טענה</u>

אם $B \leq_p A$ וגם $C \leq_p B$ אז גם $C \leq_p A$.

<u>הוכחה:</u>

לפי $B \leq_p A$ קיימת פונקציה $f$ ומ"ט דטרמיניסטית פול' $M_f$ שמחשבת את $f$ ומתקיים $w \in A \Leftrightarrow f(w) \in B$. באופן דומה לפי $C \leq_p B$ קיימת פונקציה $g$ ומ"ט דטרמיניסטית פול' $M_g$ שמחשבת את $g$ ולפיה מתקיים $w \in B \Leftrightarrow f(w) \in C$.

**בנייה** – נבנה מכונה $M_h$ שמחשבת את $h$ רדוקציה מ-$A$ ל-$C$, כך ש-$M_h$ מ"ט דטרמיניסטית פול'. עבור קלט $x$ המ"ט $M_h$ תחשב את $f(x) = y$ כמו $M_f$ ואז תחשב את $g(y) = z$ כמו $M_g$ ותחזיר את $z$. צריך להראות נכונות וזמן ריצה פולינומיאלי.

**נכונות** – $x \in A \overset{M_f}{\Leftrightarrow} y = f(x) \in B \overset{M_g}{\Leftrightarrow} z = g(y) \in C$.

**זמן ריצה** –

(א) הפעלת $M_f$ – ידוע ש-$M_f$ מ"ט פול' ולכן רצה בזמן $O(|x|^{k_1})$ עבור קבוע $k_1$ כלשהו.

(ב) הפעלת $M_g$ – ידוע ש-$M_g$ מ"ט פול' ולכן רצה בזמן $O(|y|^{k_2})$ עבור קבוע $k_2$ כלשהו.

הבעיה – זמן הריצה של שלב ב' מתואר כפונקציה של $y$ שהוא משתנה פנימי, בעוד שהגדרנו זמן ריצה על גודל הקלט, שהוא $x$. צריך חסם כפונקציה של אורך הקלט $x$.

הפתרון – נשים לב ש-$M_f$ יכולה ליצור פלט $y$ מאורך לכל היותר כמספר צעדי הריצה שלה, לכן מתקיים כי $|y| = O(|x|^{k_1})$. כלומר, זמן ריצת שלב (ב) הוא $O(|x|^{k_1 + k_2})$, אז זמן ריצת (א)+(ב) הוא $O(|x|^{k_1 + k_2})$ עם $k_1, k_2$ קבועים.

## קשיות ושלמות של מחלקות

$L \in \text{NP-Complete}$ אם"ם $L \in \text{NP}$ וגם $L \in \text{NP-hard}$.

האם $\text{P} = \text{NP}$? אם הייתה לכך הוכחה/ הפרכה, מאיזה כיוון היא תגיע?

<u>כיוון הוכחה</u>: אם $L \in \text{NP-Complete}$ וגם $L \in \text{P}$ אז $\text{P} = \text{NP}$.

<u>הסבר</u>: אם $L \in \text{NPC}$ אז $L$ היא NP-קשה, כלומר לכל $L' \in \text{NP}$ מתקיים $L' \leq_p L$ ואם בנוסף $L \in \text{P}$ לפי משפט הרדוקציה הפול"י $L' \in \text{P}$ כלומר $\text{NP} \subseteq \text{P}$, ואת הכיוון השני כבר ראינו לכן NP=P.

<u>כיוון הפרכה</u>: אם $L \in \text{NP-Complete}$ וגם $L \notin \text{P}$ אז $\text{P} \neq \text{NP}$.

## דוגמאות לשפות NP-קשות

### 3-SAT

נגדיר: $\text{3-SAT} = \{\varphi \mid \text{נוסחת 3-CNF וגם } \varphi \text{ ספיקה } \varphi\}$

<u>תזכורת</u>: נוסחה בוליאנית תיקרא CNF אם היא מהצורה הבאה – מורכבת מפסוקיות (הסגרים). בתוך כל פסוקית משתנים ושלילת משתנים ביניהם סימן ∨. בין הפסוקיות סימן ∧.

נוסחה תיקרא 3-CNF אם בכל פסוקית יש בדיוק 3 ליטרלים (3 משתנים או שלילת משתנים). נוסחה $\varphi$ תיקרא ספיקה אם"ם יש הצבה מספקת למשתנים, כלומר הצבה של ערכי True/False למשתנים כך שערך האמת של $\varphi$ הוא True.

<u>טענה</u>: $\text{3-SAT} \in \text{NP-Complete}$. צריך להראות (א) $\text{3-SAT} \in \text{NP}$ (ב) $\text{3-SAT} \in \text{NP-hard}$.

(א) הוכחה:

נראה ש-$\text{3-SAT} \in \text{NP}$. כלומר, נראה שקיימת מ"ט פול"ד שמכריעה את 3-SAT. המכונה $M$:

- תוודא שהנוסחה מהצורה 3-CNF.
- תנחש הצבה ותוודא שההצבה מספקת את $\varphi$.

זמן ריצה פול" – ברור. נכונות – אם יש הצבה מספקת אז יש ריצה מקבלת, אם לא אז אין כזו.

(ב) שבוע הבא.

### CLIQUE

נגדיר: $\text{CLIQUE} = \{\langle G, k\rangle \mid G \textit{ is an undirected graph with a clique of size } k\}$

(כאשר קליקה היא קבוצת קודקודים שמחוברים בקשת כולם אלו לאלו).

<u>טענה</u>: CLIQUE ∈ NP-Complete. נראה: (א) CLIQUE ∈ NP (ב) CLIQUE ∈ NP-hard.

(א) <u>הוכחה</u>:

נראה מ"ט ל"ד שמכריעה את CLIQUE. המכונה עבור קלט $\langle G,k\rangle$ תנחש באופן ל"ד תת-קבוצה $S$ של צמתים ואז תוודא באופן דטרמיניסטי את הבאים:

(1) תספור כמה צמתים ב-$S$, אם לא שווה $k$ תעצור ותחזיר $q_{rej}$.
(2) לכל זוג צמתים ב-$S$ תבדוק האם יש ביניהם קשת.

אם נמצא זוג ב-$S$ ללא קשת – תעצור ותחזיר $q_{rej}$. אם יש קשת בין כל זוג צמתים ב-$S$ תחזיר $q_{acc}$.

**נכונות:**

אם $\langle G,k\rangle \in$ CLIQUE אז קיימת תת-קבוצה $S$ של צמתי $G$ כך ש-$|k| = S$ וגם בין כל זוג צמתים ב-$S$ יש קשת. ולכן אם המכונה $M$ תנחש את $S$ אז תסיים במצב $q_{acc}$.

אם $\langle G,k\rangle \notin$ CLIQUE אז לכל ניחוש $S$, או שאינו בגודל $k$ או שיש בו צמתים ללא קשת ולכן בכל מקרה $M$ תחזיר $q_{rej}$.

**זמן ריצה:**

ניחוש $S$: לינארי באורך הקלט.

בדיקת קשתות: $O(k^2)$ זוגות ולכל זוג $O(n)$. סה"כ $O(n^3)$ ולכן זמן ריצה פול".

(ב) <u>הוכחה</u>:

נרצה להראות ש-CLIQUE ∈ NP-hard. כלומר לכל $L' \in$ NP יתקיים $L' \leq_p$ CLIQUE נראה "רק" רדוקציה 3-SAT $\leq_p$ CLIQUE, ואז כיוון שהוכחנו כבר 3-SAT∈NP-hard מתקיים לכל $L' \in$ NP ש-3-SAT $L' \leq_p$ הראינו שרדוקציות פול" סגורות להרכבה. לכן, נסיק שלכל $L' \in$ NP מתקיים CLIQUE $L' \leq_p$ כלומר CLIQUE ∈ NP-hard.

<u>צ"ל</u>: 3-SAT $\leq_p$ CLIQUE, כלומר שקיימת פונקציה $f$ וקיימת מ"ט $M_f$ כך ש-$M_f$ מחשבת את $f$ בזמן פול" וגם $f$ פונקציית רדוקציה. כלומר:

$$\varphi \in \text{3-SAT} \Leftrightarrow f(\varphi) = \langle G,k\rangle \in \text{CLIQUE}$$

ב-$G$ יש קליקה בגודל $k$ $\Leftrightarrow$ $\varphi$ היא 3-CNF ספיקה

**בנייה:**

בהינתן נוסחה $\varphi$ המכונה $M_f$ תפעל באופן הבא:

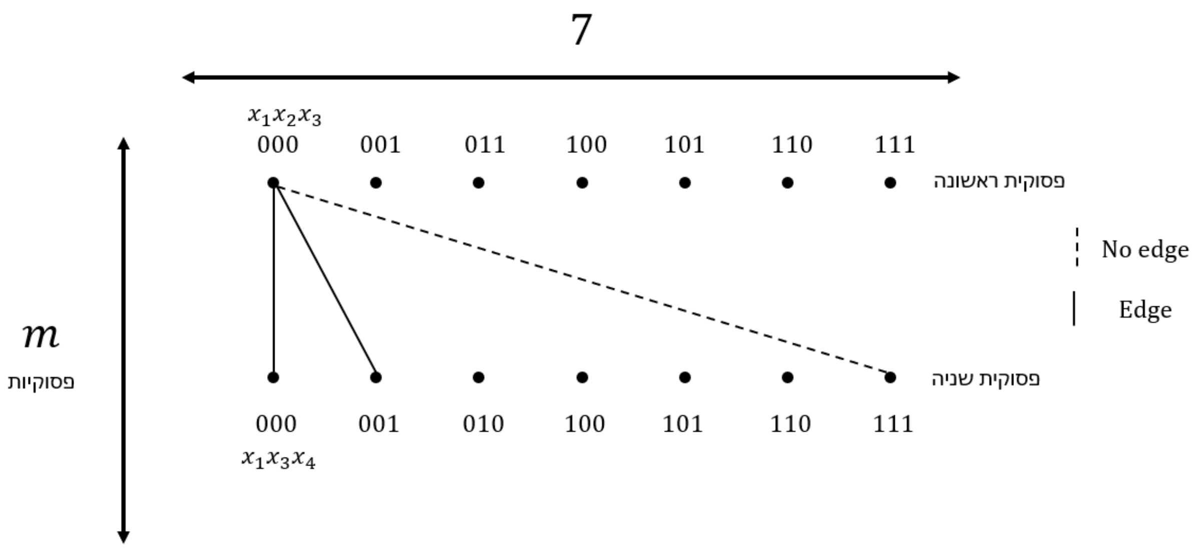
(1) תבדוק האם $\varphi$ מצורת 3-CNF (קל), אם לא, תחזיר $\langle G,k\rangle \notin$ CLIQUE (כמו למשל $k$ שגדול ממספר הצמתים בגרף ולכן כמובן אין קליקה בגודל $k$).
(2) בה"כ $\varphi$ מצורת 3-CNF. $M_f$ תפעל באופן הבא:

- לכל פסוקית ב-$\varphi$ תגדיר 7 צמתים, אחד לכל הצבה מספקת של הפסוקית.
- בין כל זוג צמתים $M_f$ תחבר קשת אלא אם הם מתייחסים לאותו משתנה ולא מסכימים בהצבה עליו.
- $k$ יהיה מספר הפסוקיות ב-$\varphi$.

הערה: בתוך שכבה לעולם לא תהיה קשת, כי מדובר באופציות שונות לאותה ההשמה.

**דוגמה:**

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4)$$




בתוך שכבה לעולם לא תהיה קשת, כי מדובר באופציות שונות לאותה ההשמה.

**זמן ריצה:**

יצירת הצמתים: זמן פולינומיאלי ב-$\varphi$ (יש $7m$ צמתים, $m < |\varphi|$)

חיבור קשתות: צריך לעבור על כל זוגות הצמתים $O((7m)^2)$ כלומר $O(m^2) = O(|\varphi|^2)$.

לכל זוג לבדוק: אם לא מתייחסים למשתנה משותף מחברים בקשת, אם כן מתייחסים למשתנה משותף ונותנים לו את אותו הערך – נחבר בקשת, אחרת לא. זה לוקח $O(|\varphi|)$ לכל זוג לכן סה"כ קיבלנו $O(|\varphi|^3)$.

**נכונות:**

בכיוון $\Leftarrow$ צ"ל שאם $\varphi \in$ 3-SAT אז $\langle G,k\rangle \in$ CLIQUE. כלומר אם $\varphi$ נוסחת 3-CNF ספיקה אז ב-$G$ יש קליקה בגודל $k$. נסתכל על הצבה מספקת $A$ עבור $\varphi$. לכל פסוקית, נבחר צומת (מבין השביעייה) שמתאים להצבה שמסכימה עם $A$. נשים לב ש-$A$ הצבה מספקת, בפרט מספקת כל אחת מהפסוקיות ולכן מתאימה לבדיוק צומת אחת מכל שביעייה.

טענה: הקבוצה $S$ של צמתים שבחרנו כך היא קליקה ב-$G$ בגודל $k$.

הוכחה: גודל $k$ – כי ב-$S$ יש בדיוק צומת אחת בכל שכבה/ פסוקית. קליקה – נבהיר למה כל זוג צמתים ב-$S$ מחוברים בקשת: נסתכל על זוג צמתים ב-$S$. אם הם לא מתייחסים למשתנה משותף לפי הבנייה בהכרח יש ביניהם קשת ואם הם כן מתייחסים למשתנ/ים משותפ/ים, כיוון שההצבה לכל פסוקית נגזרה מהצבה כללית $A$ יש הסכמה על ההצבה למשתנים משותפים, כלומר לפי הבנייה יש קשת בין זוג הצמתים.

הסבר נוסף לאינטואיציה: בכל שורה נעבור ונבחר את הקודקוד שמייצג את ההצבה הגלובלית שמספקת. לכן סה"כ נקבל $k$ צמתים. למה מחוברות בקשת? אם אין משתנה משותף יש קשת, אבל אם יש משותף תהיה קשת כי כל הפסוקיות סופקו על ידי אותה ההצבה, אז למשל עבור $x_7$ בשניהם יש True.

בכיוון $\Longrightarrow$ צ"ל שאם $\langle G, k\rangle \in \mathrm{CLIQUE}$ אז $\varphi \in 3\text{-SAT}$. נסתכל על תת קבוצה $S$ של הצמתים שהיא קליקה ב-$G$ בגודל $k$.

נשים לב ש-$S$ בהכרח מכילה בדיוק צומת אחד מכל שכבה. הסבר: לא יותר מאחד כי בתוך שכבה אין קשתות, ו-$S$ היא קליקה. לא פחות מאחד כי גודל הקליקה כמספר השכבות.

נגדיר הצבה $A$ בעזרת $S$. לכל פסוקית – הצומת ב-$S$ מהשכבה המתאימה מגדיר הצבה למשתני הפסוקית. נשים לב שזוג צמתים שמתייחסים למשתנה משותף מסכימים על ההצבה עליו כי $S$ קליקה, כלומר הם מחוברים בקשת לכן קיבלנו הצבה תקנית לכל המשתנים. בנוסף, כל צומת מתאים להצבה של אותה ההשמה, וכיוון שב-$S$ יש צומת מכל שכבה $A$ מספקת את כל הפסוקיות כלומר מספק את $\varphi$ כלומר $\varphi \in 3\text{-SAT}$.

## מוודא פולינומי

נציג הגדרה חלופית ל-NP.

**הגדרה – מוודא פולינומי:** מ"ט דטרמיניסטית $\boldsymbol{M}$ היא מוודא פולינומי עבור שפה $\boldsymbol{L}$ אם $\boldsymbol{M}$ מקבלת זוג קלטים $(\boldsymbol{w}, \boldsymbol{c})$ רצה בזמן פולינומי ב-$|\boldsymbol{w}|$, ומתקיים:

לכל $\boldsymbol{w} \in \boldsymbol{L}$ קיים $\boldsymbol{c}$ עבורו $\boldsymbol{M}(\boldsymbol{w}, \boldsymbol{c}) = \boldsymbol{q_{acc}}$

ולכל $w \notin L$ ולכל $c$ מתקיים $M(w, c) = q_{rej}$.

**הגדרה (חלופית) – $\mathrm{NP}'$:** נגדיר $\mathrm{NP}'$ להיות מחלקת כל השפות $\boldsymbol{L}$ שיש עבורן מוודא פולינומי.

טענה

$$\mathrm{NP} = \mathrm{NP}'$$

<u>דוגמה</u>:

נראה שיש מוודא פולינומי עבור השפה CLIQUE.

$w$ יהיה זוג $\langle G,k\rangle$ המועמד לשייכות ל-CLIQUE .

$c$ יהיה העד לשייכות, במקרה זה אמור לייצג קליקה ב-$G$ בגודל $k$.

המ"ט הדטרמיניסטית $M$ תבדוק האם $c$ הוא קבוצה של $k$ צמתים, וכן האם בין כל שניים מהם יש קשת. אם כן – $M$ תחזיר $q_{acc}$, ואחרת $q_{rej}$.

<u>נכונות</u>: אם $\langle G,k\rangle \in$ CLIQUE ההצבה של $c$ להיות קליקה בגודל $k$ תגרום ל-$M$ להחזיר $q_{acc}$ ואם $\langle G,k\rangle \notin$ CLIQUE אז כל הצבה של $c$, לא משנה מה תהיה, תסתיים ב-$q_{rej}$.

<u>זמן ריצה</u>: קל לראות שפולי" ב-$|\langle G,k\rangle|$.

# שבוע 9 – הרצאה

## הוכחת הטענה $\mathrm{NP} = \mathrm{NP'}$

הערה: בהרצאה הוכחה זו הוצגה בסוף השיעור, אך על מנת לשמור על הרצף עם סוף השיעור הקודם הקדמתי אותה לכאן.

נוכיח בעזרת הכלה דו כיוונית:

<u>כיוון ראשון $\mathrm{NP} \supseteq \mathrm{NP'}$</u>

נתונה שפה $L$ ויש עבורה מוודא פולינומי דטרמיניסטי, כלומר $M'$ בהתאם להגדרות שראינו לעיל. נרצה בעזרת $M'$ לבנות מכונה $M$ שהיא פולינומית לא דטרמיניסטית ומזהה את $L$. בה"כ האורך של העד $c$ חסום על ידי פולינום ב-$|w|$. הסבר: $M'$ רצה בזמן פול" ב-$|w|$ ולכן לא יכולה להספיק לקרוא יותר ממספר פולינומי של אותיות של $c$.

**בנייה:** $M$ תפעל באופן הבא – תייצר מחרוזת $c$ באורך פולינומי ב-$|w|$ באופן לא דטרמיניסטי, ואז תרוץ כמו $M'$ על $w, c$.

**נכונות:**

- אם $w \in L$ אז יש $c$ עבורו $M'(w, c) = q_{acc}$ ולכן יש ניחוש של $M$ עבורו ערכי האותיות של $c$ יגרום ל-$M$ להחזיר $q_{acc}$.
- אם $w \notin L$ אז לכל $c$ מתקיים $M'(w, c) = q_{rej}$ ולכן לכל $c$ ש-$M$ תייצר הריצה תסתיים במצב $q_{rej}$.

**זמן ריצה:** לייצר את $c$ לוקח אורך של $c$, כלומר בה"כ חסום מלמעלה על ידי פולינום ב-$|w|$. בנוסף, השלב של הרצת $M'(w, c)$ ידוע שלוקח זמן פולינומי ב-$|w|$, סה"כ זמן ריצה פול" ב-$|w|$.

<u>כיוון שני $\mathrm{NP} \subseteq \mathrm{NP'}$</u>

נתונה שפה $L$ ויש עבורה מ"ט פול" ל"ד $M$. נרצה בעזרתה לבנות מ"ט $M'$ שהיא מוודא פולינומי עבור השפה $L$. נתבונן בפונקציית המעברים $\delta$, מה הטווח שלה?

$$\delta: \ Q \times \Gamma \rightarrow \mathrm{P}[Q \times \Gamma \times \{R, L\}]$$

מספר האפשרויות $k$ לבחירה לא דטרמיניסטית בכל צעד וצעד הוא לכל היותר $k \coloneqq |Q| \cdot |\Gamma| \cdot 2$. נסתכל על ריצה של $M(w)$. בכל צעד ריצה יש בחירה ל"ד של $\delta$. זה מגדיר מחרוזת שבה כל אות היא מספר בין 1 ל-$k$. אורך המחרוזת הוא כמספר צעדי הריצה, כלומר פול" ב-$|w|$.

**בנייה:** אפשר להגדיר מ"ט $M'$ דטרמיניסטית שמקבלת קלט $w$ ומחרוזת כזו $c$ ומדמה את ריצת $M(w)$ כאשר החלופה לצעד ל"ד זה צעד שנבחר על פי אות מתוך $c$. כלומר $M'$ פועלת כמו $M$ בכל צעד מדומה של $M$ היא קוראת אות נוספת מתוך $c$ ובעזרתה בוחרת איזה צעד של $\delta$ (של $M$) לבצע. נשים לב ש-$M'$ דטרמיניסטית.

**נכונות:**

- אם $w \in L$ קיימת ריצה מקבל של $M(w)$ כלומר קיימת סדרה של בחירות ל"ד של $\delta$ שגורמות ל-$M$ להחזיר $q_{acc}$. לכן, אם $c$ תהיה המחרוזת שמתארת את סדרת הבחירות האלה אז $M'(w,c)$ תחזיר $q_{acc}$.
- אם $w \notin L$ אז כל ריצה של $M(w)$ מחזירה $q_{rej}$, לכן כל מחרוזת $c$ תוביל ל-$M'(w,c) = q_{rej}$.

**זמן ריצה:** האורך של $c$ הוא פולינומי ב-$|w|$. מספר צעדי הסימולציה ש-$M'$ עושה הוא כמספר הצעדים של $M(w)$, כלומר פולינומי ב-$|w|$. סימולציה של כל צעד לוקחת לכל היותר כאורך הסרט, כלומר לכל היותר זמן פולינומי ב-$|w|$. סה"כ זמן ריצה פולינומי ב-$|w|$.

## משפט קוק-לוין

ראינו בעבר $\text{SAT} \in \text{NP}$, $\text{3-SAT} \leq_p \text{SAT}$. לכן:

$$\text{SAT} \in \text{NP-Complete} \Longrightarrow \text{3-SAT} \in \text{NP-Complete}$$

צריך להוכיח שלכל $L \in \text{NP}$ אם $L \leq_p \text{SAT}$ אז $w \in L \Leftrightarrow \varphi \in \text{SAT}$. מה הבעיה? $L$ לא ידועה. כל שידוע לנו זה שהיא נמצאת ב-NP, כלומר שקיימת מ"ט $M$ פול' לא דטרמיניסטית שמזהה את $L$.

<u>תזכורת – קונפיגורציה</u>

קונפיגורציה היא תיאור של מצב כללי במהלך ריצה. פורמלית $C = uqv$ כאשר $q \in Q$, $u, v \in \Gamma^*$. הקונפיגורציה $c$ מתארת מצב כללי בו המצב הפנימי של המכונה הוא $q$, תוכן הסרט הוא $uv$ והראש על האות הראשונה של $v$.

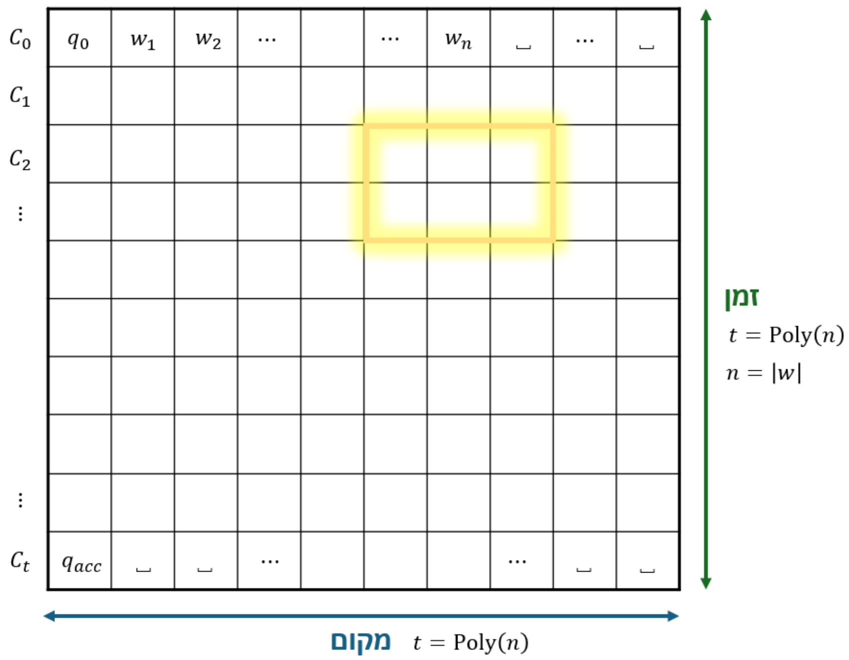
<u>פירוט נוסף</u>: הקונפיגורציה ההתחלתית בריצת $M(w)$ היא $C_0 = q_0 w$ (לשים לב – היא <u>יחידה</u>). קונפיגורציה מקבלת היא קונפיגורציה שמכילה את המצב $q_{acc}$. קונפיגורציה דוחה היא קונפיגורציה שמכילה את המצב $q_{rej}$.

**טענת עזר:** בה"כ, למכונה $M$ יש קונפיגורציה מקבלת יחידה.

**הוכחה:** נראה איך לשנות את $M$ כך שתהיה לה קונפיגורציה מקבלת יחידה וזמן הריצה יישאר פולינומי. נשנה את מצב $q_{acc}$ למצב $q_{acc}^{almost}$. במצב זה $M$ מוחקת את כל הסרט, מחזירה את הראש לתחילת הסרט ועוברת למצב $q_{acc}$. ברור שעכשיו יש קונפיגורציה מקבלת יחידה, ונשים לב שזמן הריצה הוארך בלכל היותר פולינום באורך הקלט כי על הסרט שהיה צריך למחוק יש לכל היותר מספר פולינומי בגודל הקלט של תווים.

באופן דומה, מוכיחים של-$M$ בה"כ יש קונפיגורציה דוחה יחידה.

<u>רעיון ההוכחה</u>: נתאר ריצה של $M$ על $w$ על ידי טבלה. כל שורה בטבלה מתאימה לקונפיגורציה, החל משורה 0 בה יש את $C_0$, (קונפ' התחלתית). ריצה תתאים למילוי חוקי של הטבלה, ואם השורה האחרונה היא הקונפיגורציה המקבלת אז המילוי מתאים לריצה מקבלת. נבנה נוסחה שבודקת האם המילוי בטבלה חוקי וכן מסתיים בקונפיגורציה מקבלת.




כל תא בטבלה מכיל אות מתוך $\Gamma \cup Q$. גובה הטבלה $t$ לפי חסם על זמן הריצה. רוחב הטבלה לפי חסם על אורך הקונפיגורציה הארוכה ביותר, כלומר חסם על המקום ולכן גם חסום על ידי $t$. לחלק המסומן בצהוב בטבלה נקרא לאורך ההוכחה "שישייה".

<u>מילוי הטבלה הוא חוקי אם</u>:

- בשורה 0 מופיעה הקונפיגורציה ההתחלתית.
- בשורה $t$ מופיעה קונפיגורציה מסיימת (מקבלת אם הריצה מקבלת).
- לכל $0 \leq i < t$ מתקיים $C_i, C_{i+1}$ קונפיגורציות עוקבות, כלומר מתאימות לצעד ריצה אפשרי של $M$. $C_i, C_{i+1}$ זהות כמעט בכל מקום למעט אולי במקום בו מסומן מיקום הראש של המכונה, ושם ההבדלים צריכים להתאים לפונקציה $\delta$ של $M$.

<u>העקרון ברור, מה נותר להראות?</u>

(א) איך ננסח את כל הבדיקות של השורה ה-0, השורה ה-$t$ ושל כל השישיות?

(ב) איך להמיר את כל הנ"ל לנוסחאות CNF, בפרט מעל משתנים בוליאנים?

(ג) איך לייצר את נוסחת ה-CNF ע"י מ"ט $M'$ פולי דטרמיניסטית? (מכונת הרדוקציה).

(ד) להוכיח נכונות וזמן ריצה.

### (א) בניית בדיקות נכונות עבור מילוי של הטבלה

נקרא לתאים בטבלה $x_{i,j}$ כאשר $x_{0,0}$ הוא התא השמאלי עליון, $x_{t,t}$ התא הימני תחתון.

- <u>בדיקת שורה 0</u>

$$\varphi_{init}(x_{0,0}, x_{0,1}, \dots, x_{0,t}) = (x_{0,0} = q_0) \wedge \bigwedge_{j=1}^{n} (x_{0,j} = w_j) \wedge \bigwedge_{j=n+1}^{t} (x_{0,j} = \sqcup)$$

נשים לב שאם בהמשך כל תנאי ייבדק על ידי נוסחת CNF אז $\varphi_{init}$ כוללה תהיה נוסחת CNF.

- <u>בדיקת שורה $t$</u>

$$\varphi_{acc}(x_{t,0}, \dots, x_{t,t}) = (x_{t,0} = q_{acc}) \wedge \bigwedge_{j=1}^{t} (x_{t,j} = \sqcup)$$

- <u>בדיקת שישיות</u>

נרצה דרך לבדוק ששתי שורות עוקבות מתאימות לקונפיגורציות עוקבות.

אפשרויות למילוי חוקי של שישייה:

$\forall a, b, c \in \Gamma$

| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $a$ | $b$ | $c$ |

אילו מילויים חוקיים יש כאשר האזור עובר שינוי (הראש בסביבה)? עבור $\delta(q_1, a) \ni (q_2, b, R)$

| | | |
|---|---|---|
| $q_1$ | $a$ | $c$ |
| $b$ | $q_2$ | $c$ |

| | | |
|---|---|---|
| $c$ | $q_1$ | $a$ |
| $c$ | $b$ | $q_2$ |

| | | | |
|---|---|---|---|
| $d$ | $c$ | $q_1$ | $a$ |
| $d$ | $c$ | $b$ | $q_2$ |

| | | | |
|---|---|---|---|
| $q_1$ | $a$ | $c$ | $d$ |
| $b$ | $q_2$ | $c$ | $d$ |

$\forall c, d \in \Gamma$

באופן דומה נוסיף את כל הבדיקות עבור מילוי חוקי לשישייה כאשר פקודת התנועה היא שמאלה. נגדיר בדיקה עבור שישייה:

$$\varphi_{legal}(x_{i,j}, x_{i,j+1}, x_{i,j+2}, x_{i+1,j}, x_{i+1,j+1}, x_{i+1,j+2})$$

בודקת האם המילוי חוקי לפי לפחות אחת מהאפשרויות שתיארנו קודם.

קיבלנו שהנוסחה היא:

$$\varphi = \varphi_{init} \wedge \varphi_{acc} \wedge \bigwedge_{i=0}^{t} \bigwedge_{j=0}^{t} \varphi_{legal}$$

**(ב) המרה לנוסחאות CNF מעל משתנים בוליאנים**

טסט $\varphi_{legal}$ מחזיר $T$ אם"ם ההצבה לששת המשתנים הרלוונטיים היא הצבה מותרת מתוך כלל ההצבות שהן $(\Gamma \cup Q)^6$.

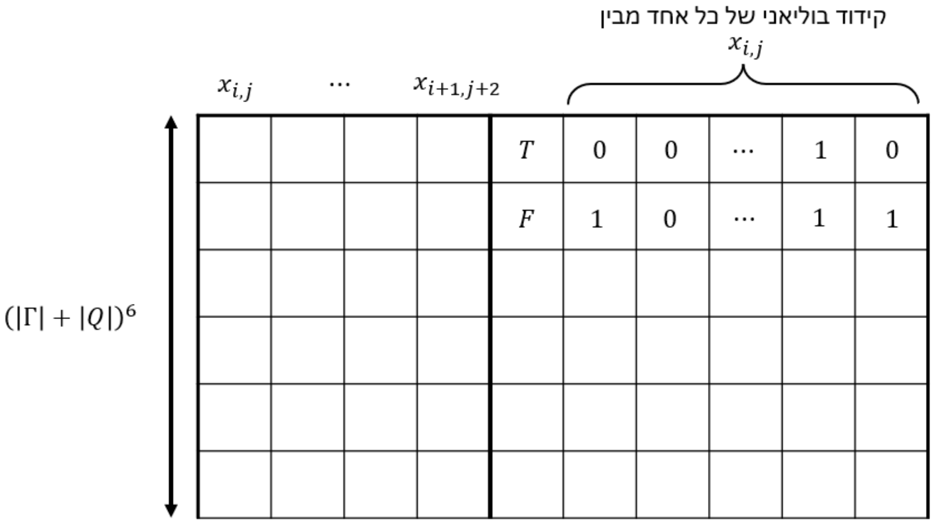
נרצה להעביר הכל לעולם בוליאני, כלומר כל משתנה בטבלה $x_{i,j}$ יוחלף על ידי קבוצת משתנים בוליאנים.

כל בדיקה נרצה לנסח אותה: **(1)** כנוסחה על משתנים בוליאניים **(2)** בצורת CNF

לכל משתנה מקורי $x_{i,j}$ נצטרך $\lceil \log_2(|\Gamma| + |Q|) \rceil$ משתנים בוליאנים. זה קבוע שתלוי במכונה – לא בשפה. נסמן ב-ˆ את הגרסה הבוליאנית של הנוסחאות מהשלב הקודם: $\varphi_{init}(x_{0,0}, \dots, x_{0,t})$. מאלצת ערך אחד ויחיד לכל משתנה שלה, ולכן גם $\hat{\varphi}_{init}$ עושה אותו דבר. ... $\wedge \left(\overline{x^2}_{0,0}\right) \wedge \left(x^1_{0,0}\right)$

באופן דומה גם את $\hat{\varphi}_{acc}$ קל לתאר כנוסחת CNF.

נותר לטפל ב-$\varphi_{legal}$. תהיה עמודה לכל משתנה בוליאני שמקודד כל אחד מהשישייה המקורית. לכל הצבה לשישיית משתנים מקוריים יש הצבה מתאימה למשתנים הבוליאניים שמקודדים אותם. בבל שורה כזאת, נרצה שהנוסחה תחזיר $T$ אם הערך לשישייה חוקי ו-$F$ אחרת. כלומר, קיבלנו את טבלת האמת של הנוסחה $\hat{\varphi}^{i,j}_{legal}$ ונרצה למצוא ייצוג CNF שמסכים עם טבלת האמת הזו. אם לא הייתה דרישה לצורת CNF, פתרון אפשרי הוא נוסחת DNF, כלומר לכל שורה שמתאימה לערך (פלט) $T$, נשים פסוקית שבתוכה $\wedge$ בין משתנים, ומאלצת את ערכי האמת המתאימים לשורה (ההצבה) ובין הפסוקיות $\vee$.




כדי לייצר נוסחת CNF נייצר נוסחת DNF עבור כל השורות המסומנות ב-$F$, ואז שלילה של הנוסחה המתקבלת (דה-מורגן) תיתן CNF עבור $\hat{\varphi}^{i,j}_{legal}$.

$$\hat{\varphi} = \hat{\varphi}_{init} \wedge \hat{\varphi}_{acc} \wedge \bigwedge_{i=0}^{t} \bigwedge_{j=0}^{t} \hat{\varphi}^{i,j}_{legal}$$

סיימנו להראות שקיימת נוסחת CNF שסימנו ב-$\hat{\varphi}$, והיא ספיקה אם"ם $w \in L$.

### (ג) ייצור הנוסחה על ידי מ"ט $M$ דטרמיניסטית פולינומיאלית

צריך להראות שקיימת רדוקציה $L \leq_p \text{SAT}$ כלומר שקיימת מ"ט $M$ דטרמיניסטית פולינומיאלית שעבור קלט $w$ מחזירה $\hat{\varphi}$ כך ש-$w \in L \Leftrightarrow \varphi \in \text{SAT}$, כלומר להראות איך מייצרים את $\hat{\varphi}$ מ-$w$ ע"י מ"ט $M$ דטרמיניסטית פולינומיאלית.

<u>בנייה:</u>

(1) $M'$ תחשב את $t$ (פולינום כלשהו על $|w| = n$), קל לבצע בזמן פול".

(2) $M'$ תייצר את $\varphi'_{init}$ וגם את $\varphi'_{acc}$, קל לבצע בזמן פול".

(3) $M'$ תריץ לולאה כפולה (על $i$ ועל $j$, מ-0 ועד $t$) ולכל $i, j$ תדפיס עותק של $\hat{\varphi}^{i,j}_{legal}$. נשים לב שכל הנוסחאות $\hat{\varphi}^{i,j}_{legal}$ הן אותו הדבר עד כדי שינוי שמות משתנים. כלומר, הנוסחה $\hat{\varphi}_{legal}$ יכולה להיות חלק מקידוד המכונה $M'$, ו-$M'$ רק צריכה להדפיס אותה שוב ושוב עם עדכון $i, j$ ובין כל שני עותקים סימן $\wedge$.

<u>הערה</u>: הפולינום שמשמש לצורך חישוב $t$ מתוך $|w| = n$ קיים, ולכן יכול להיות מקודד בתוך $M'$.

### (ד) הוכחת נכונות וזמן ריצה

<u>זמן ריצה</u>: $M'$ צריכה להדפיס את $\hat{\varphi}_{init}$, $\hat{\varphi}_{acc}$ כל אחת בזמן פול". בנוסף, היא צריכה להדפיס את $\hat{\varphi}^{i,j}_{legal}$ לכל ערך $i, j$ בין 0 ל-$t$ פעמים, כלומר $t^2$ פעמים. סה"כ מספר פול" של פעמים, ולכן זמן הריצה כולו הוא פולינומיאלי.

<u>נכונות:</u>

- אם $w \in L$ אז קיימת ריצה מקבלת של $M(w)$ ולכן קיים מילוי חוקי לטבלה המקורית שבו שורה ראשונה היא $C_0$ (הקונפיגורציה ההתחלתית), שורה אחרונה היא קונפיגורציה מקבלת וכל 2 שורות עוקבות מתאימות לקונפיגורציה עוקבת בריצה – וכל הבדיקות יעברו. לכן, קיימת הצבה למשתנים בוליאנים שבה כל בדיקות ה-CNF עוברות, ולכן יש הצבה מספקת עבור $\varphi$.
- אם $w \notin L$ אז כל מילוי של הטבלה המקורית הוא או מתאים לריצה חוקית שמסתיימת בקונפיגורציה הדוחה, או אינו מתאים לריצה חוקית. בכל מקרה, לפחות בדיקה אחת מתוך $\varphi_{init}$, $\varphi_{acc}$, $\varphi^{i,j}_{legal}$ נכשלת, לכן לפחות נוסחת CNF אחת מבדיקות המשתנים הבוליאנים מחזירה $F$ ולכן כל הצבה לא מספקת את $\hat{\varphi}$. מסקנה $\hat{\varphi} \notin \text{SAT}$.

# שבוע 11 – הרצאה

## משפט ההיררכיה

### משפט ההיררכיה בזמן

לכל פונקציה $f$ שחשיבה בזמן $O\left(f(n)\right)$ מתקיים:

$$\text{Time}\left(o\left(f(n)\right)\right) \subsetneq \text{Time}\left(f(n)\log n\right)$$

### משפט ההיררכיה במקום

לכל פונקציה $f$ שחשיבה במקום $O\left(f(n)\right)$ מתקיים:

$$\text{Space}\left(o\left(f(n)\right)\right) \subsetneq \text{Space}\left(f(n)\right)$$

<u>הרעיון</u>: נחזור על הוכחת הלכסון שמראה $\text{R}\subsetneq\text{RE}$ ואז נעדכן אותה עם מגבלה על זמן/מקום ריצה, ונקבל את משפט ההיררכיה.

<u>תזכורת להוכחת הלכסון</u>: $A_{TM} = \{\langle M,w\rangle \mid M(w) = q_{acc}\}$.

נניח בשלילה שקיימת מ"ט דטרמיניסטית $H$ שמכריעה את $A_{TM}$, כלומר:

$$H(\langle M,w\rangle) = q_{acc} \Leftrightarrow M(w) = q_{acc}$$

נגדיר:

$$D(\langle M\rangle) = H(\langle M\rangle,\langle M\rangle) \quad ; \quad \underbrace{\widehat{D}(\langle M\rangle) = \overline{D(\langle M\rangle)}}_{swap\ q_{acc}\ and\ q_{rej}}$$

מה מחזירה $\widehat{D}\left(\langle\widehat{D}\rangle\right)$?

אם $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{acc}$ אז $D\left(\langle\widehat{D}\rangle\right) = q_{rej}$ ולכן $H\left(\langle\widehat{D}\rangle,\langle\widehat{D}\rangle\right) = q_{rej}$ אז $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{rej}$ סתירה.

אם $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{rej}$ אז $D\left(\langle\widehat{D}\rangle\right) = q_{acc}$ ולכן $H\left(\langle\widehat{D}\rangle,\langle\widehat{D}\rangle\right) = q_{acc}$ אז $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{acc}$ סתירה.

### הוכחה – זמן:

נגדיר שפה:

$$A_f = \{\langle M,w\rangle \mid M(w) = q_{acc} \land M's\ run\ on\ w\ finishes\ with\ at\ most\ f(|w|)\ steps\}$$

ניתן להכריע את $A_f$ בעזרת מ"ט אוניברסלית ומונה (בכמה זמן?) נראה בדומה להוכחה עבור $A_{TM}$ שלא ניתן להכריע את $A_f$ בזמן $o\left(f(n)\right)$.

נניח בשלילה שקיימת מ"ט $H_f$ שמכריעה את $A_f$ תוך הסתפקות ב-$o\left(f(n)\right)$ זמן ריצה.

נבנה בעזרת $H_f$ את המ"ט $D$: $D(\langle M\rangle) = H_f(\langle M\rangle, \langle M\rangle)$.

נבנה בעזרת $D$ את המ"ט $\widehat{D}$: $\widehat{D}(\langle M\rangle) = \overline{D(\langle M\rangle)}$.

מה מחזירה $\widehat{D}\left(\langle\widehat{D}\rangle\right)$?

אם $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{acc}$ אז $D\left(\langle\widehat{D}\rangle\right) = q_{rej}$ ולכן $H_f\left(\langle\widehat{D}\rangle, \langle\widehat{D}\rangle\right) = q_{rej}$ אז $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{rej}$ סתירה. (זמן הריצה של $H_f$ הוא $o\big(f(n)\big)$ לכן גם של $D, \widehat{D}$ – כלומר הדחייה אינה בעקבות זמן הריצה אלא בעקבות $D\left(\langle\widehat{D}\rangle\right) = q_{rej}$).

אם $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{rej}$ אז $D\left(\langle\widehat{D}\rangle\right) = q_{acc}$ ולכן $H_f\left(\langle\widehat{D}\rangle, \langle\widehat{D}\rangle\right) = q_{acc}$ אז $\widehat{D}\left(\langle\widehat{D}\rangle\right) = q_{acc}$ סתירה.

הראינו שקיימת שפה $A_f$ שלא שייכת ל-$\text{Time}\left(o\big(f(n)\big)\right)$. ברור ש-$A_f$ כריעה, השאלה באיזו מגבלת זמן.

דרך להכריע את $A_f$: לאתחל מונה ל-$f(|w|)$ ולהריץ אז המכונה האוניברסלית $\mathcal{U}$: $\mathcal{U}(\langle M, w\rangle)$ ואחרי סימולציית כל צעד ריצה של $M$ להוריד את המונה ב-1. אם המונה הגיע ל-0, נעצור ונחזיר $q_{rej}$ ואם לפני שהמונה הגיע ל-0 הסימולציה של $M$ מגיעה ל-$q_{acc}$ או ל-$q_{rej}$ נענה בהתאם.

ברור שהמכונה הנ"ל מכריעה את $A_f$, אך בכמה זמן? יש מימוש של מכונת טיורינג אוניברסלית $\mathcal{U}$ שמדמה ריצה של $f(n)$ צעדים בעזרת $O(f(n)\log f(n))$ צעדים של $\mathcal{U}$. נדרוש שהפונקציה $f$ תהיה חשיבה בזמן $O\big(f(n)\big)$. כלומר בהינתן $1^n$ ניתן לחשב את $1^{f(n)}$ בזמן $O\big(f(n)\big)$.

**מסקנה:** לכל פונקציה $f$ שחשיבה בזמן $O\big(f(n)\big)$ מתקיים:

$$\text{Time}\left(o\big(f(n)\big)\right) \subsetneq \text{Time}\big(f(n)\log\big(f(n)\big)\big)$$

**דוגמה:**

$n^{2.1}$ חשיבה בזמן $O(n^{2.1})$ וגם $n^2 = o(n^{2.1})$ וכן $n^{2.1}\log n^{2.1} = O(n^3)$.

מסקנה: $\text{Time}(n^2) \subsetneq \text{Time}(n^3)$.

מסקנה: $\text{P} \subsetneq \text{EXP}$.

למשל, יש שפה $L \in \text{Time}(2^n)$ אבל $L \notin \text{Time}(n^k)$ לאף $k$ קבוע.

## הוכחה - מקום

ההוכחה במעט זהה להוכחה עבור משפט ההיררכיה בזמן. נחליף את מגבלת הזמן בכל נקודה בהוכחה במגבלת מקום. המשפט חזק יותר לגרסת המקום (הפרדה צפופה יותר) כי קיימת מ"ט אוניברסלית $\mathcal{U}$ שמדמה ריצה של $M'(w)$ במקום $O\big(f(n)\big)$ כאשר $f(n)$ מגבלת מקום ריצת $M$.

## משפט Savitch

לכל פונקציה $f(n)$ שחשיבה במקום $O(f(n))$.

$$\text{NSpace}(f(n)) \subseteq \text{Space}(f^2(n))$$

כל מה שניתן להכריע על ידי מ"ט ל"ד ע"י הסתפקות ב-$f(n)$ מקום, אפשר להכריע עם מ"ט דטרמיניסטית ריבועי במקום.

**מסקנה:** $\text{NPSpace} = \text{PSpace}$

תזכורת:

$$\text{PSpace} = \bigcup_{k=1}^{\infty} \text{Space}\left(n^k\right) \quad ; \quad \text{NSpace} = \bigcup_{k=1}^{\infty} \text{NSpace}\left(n^k\right)$$

**הוכחת המסקנה:**

- כיוון $\text{PSpace} \subseteq \text{NPSpace}$ כי כל מ"ט דטרמיניסטית היא מקרה פרטי של מ"ט ל"ד עם אותה מגבלת מקום.
- נראה הכלה בכיוון השני. תהי $L \in \text{NPSpace}$, כלומר קיים $k$ קבוע כלשהו כך ש-$L \in \text{NSpace}(n^k)$. לפי משפט Savitch מתקיים $L \in \text{Space}((n^k)^2)$. סך הכל קיבלנו:
$$L \in \text{Space}((n^k)^2) = \text{Space}(n^{2k}) \subseteq \text{PSpace}$$

נשים לב שאילו היה משפט תואם עבור מחלקות זמן אז היינו מסיקים $\text{P} = \text{NP}$.

הוכחת משפט Savitch

עבור מ"ט $M$ וקלט $w$ נגדיר את גרף הקונפיגורציות $G_{M,w}$. צמתי הגרף הם כל הקונפיגורציות האפשריות. יש קשת מ-$u$ ל-$v$ אם"ם הקונפיגורציה המתאימה ל-$v$ היא קונפיגורציה עוקבת של הקונפיגורציה המתאימה ל-$u$.

עבור מ"ט דטרמיניסטית לכל קונפיגורציה יש קונפיגורציה עוקבת יחידה ולכן לכל צומת יש קשת יוצאת אחת ויחידה, אבל במ"ט לא דטרמיניסטית יתכנו מספר קונפיגורציות עוקבות לכל קונפיגורציה, ולכן לכל צומת תתכן יותר מקשת יוצאת אחת.

נגדיר שני צמתים מיוחדים בגרף – צומת $C_0 = q_0 w$ שמתאים לקונפיגורציה ההתחלתית, וצומת $C_{acc}$ שמתאים לקונפיגורציה המקבלת $C_{acc} = q_{acc}$.

תזכורת: הראינו במהלך משפט קוק-לוין שבה"כ יש קונפיגורציה מקבלת יחידה. נשים לב שהשינוי ל-$M$ כדי שתהיה קונפיגורציה מקבלת יחידה לא הגדיל את סיבוכיות המקום.

**נשים לב:** $w \in L \Leftrightarrow$ קיימת ריצה מקבלת של $M(w) \Leftrightarrow$ בגרף $G_{M,w}$ יש מסלול מכוון מ-$C_0$ ל-$C_{acc}$.

כמה צמתים יש ב-$G_{M,w}$ (כמה קונפיגורציות אפשריות יש)?

$$\#\text{conf} = \underbrace{|Q|}_{\text{מצב פנימי}} \times \underbrace{O\big(f(n)\big)}_{\text{מקום ראש}} \times \underbrace{|\Gamma|^{O(f(n))}}_{\text{תוכן סרט}}$$

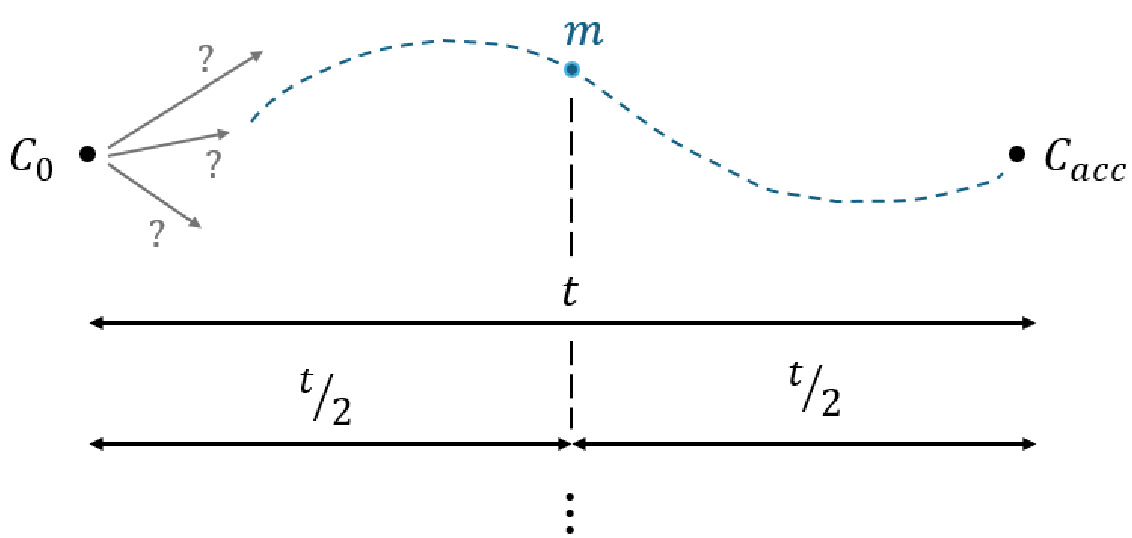
ברור שאפשר להכריע את $L$ על ידי מ"ט דטרמיניסטית באופן הבא:

נייצר את $G_{M,w}$, נריץ BFS/DFS החל מהצומת $C_0$ ונענה $q_{acc}$ אם"ם מגיעים ל-$C_{acc}$. הנכונות ברורה.

מקום הריצה $\geq$ מספר הצמתים ב-$G_{M,w}$, בפרט אקספוננציאלי ב-$f(n)$.

עבור שפה $L \in \text{NSpace}\big(f(n)\big)$ קיימת מ"ט לא דטרמיניסטית $M$ שמסתפקת במקום $O\big(f(n)\big)$. נרצה להראות שקיימת מ"ט $M'$ דטרמיניסטית שמכריעה את $L$ תוך הסתפקות במקום $O\big(f^2(n)\big)$. רוצים לבדוק אם יש מסלול בגרף הבא בלי לייצר את כולו (אחרת נחרוג מהמגבלות שהצבנו):

$G_{M,w}$




אבל, נבצע $reuse$ במקום – נזכור רק מה התשובה של השלב הקודם ואז נשתמש באותו זיכרון. הפרוצדורה $Reach(u,v,t)$ עונה על השאלה: האם יש מסלול ב-$G_{M,w}$ שמתחיל ב-$u$ מסתיים ב-$v$ ואורכו $\leq t$. נראה מ"ט דטרמיניסטית שמחשבת את $Reach$.

<u>פסאודו קוד:</u>

$\boldsymbol{Reach}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{t})$

$u = C_0, v = C_{acc}, t = \#conf$

$if\ (u = v)\ return\ \boldsymbol{T}$

$if\ (v\ is\ a\ successor\ of\ u)\ return\ \boldsymbol{T}$

$if\ (t \leq 1)\ return\ \boldsymbol{F}$

$for\ each\ m \in V\ \left(G_{M,w} = \langle V, E\rangle\right)\ do:$

$\quad q_1 = Reach\left(u, m, \left\lceil {}^{t}/_{2} \right\rceil\right)$

$\quad q_2 = Reach\left(m, v, \left\lceil {}^{t}/_{2} \right\rceil\right)$

$\quad if\ (q_1 \wedge q_2)\ return\ \boldsymbol{T}$

$return\ \boldsymbol{F}$

נכונות ברורה. <u>ננתח מקום ריצה:</u>

לכל קריאה רקורסיבית נפתח שלשה חדשה אבל 2 קריאות באותו עומק רקורסיה משתמשות באותו המקום.

נרצה לנתח את עלות המקום של $M'$. בדיקה האם $u = v$ לא דורשת מקום נוסף. בדיקה האם $v$ עוקב של $u$ גם לא דורשת מקום נוסף.

<u>כמה מקום דורשת שלשה אחת?</u>

- $u$ - $\log(\#\text{conf})$ כלומר:

$$\log\left(|Q| \times O\left(f(n)\right) \times |\Gamma|^{O\left(f(n)\right)}\right) = \log|Q| + O\left(\log\left(f(n)\right)\right) + O\left(f(n)\right) \cdot \log|\Gamma| = O\left(f(n)\right)$$

- $v$ – תיאורו דורש $O\left(f(n)\right)$
- $t$ – תיאורו דורש $O\left(f(n)\right)$

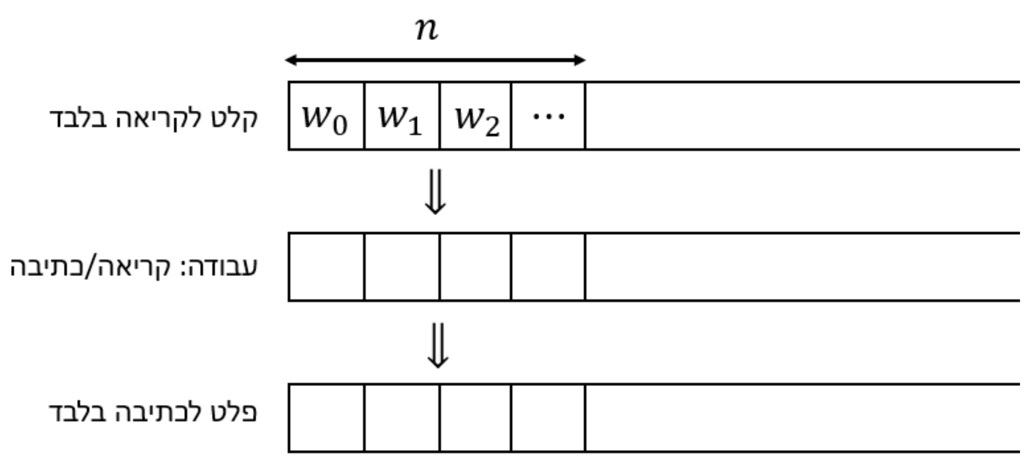
סה"כ לכל השלשה - $O\left(f(n)\right)$. מספר השלשות המקסימלי הוא עומק הרקורסיה המקסימלי (כמה שלשות יש במקביל על הסרט). סה"כ $O(\log(\#\text{conf}))$ כלומר $O\left(f(n)\right)$. סה"כ מקום $O\left(f^2(n)\right)$.

## סיבוכיות מקום תת-לינארית

<u>דוגמאות</u>: $L = \text{Space}(\log(n))$, $NL = \text{NSpace}(\log(n))$.

רלוונטי במקרים בהם זיכרון המחשב $\ll$ גודל הקלט (למשל, אפליקציה ששואלת שאלות על האינטרנט כולו).

נגדיר מודל מ"ט שמתאים לדיון על מקום תת-לינארי. מכונה עם 3 סרטים:




ראש סרט הפלט יכול בכל צעד ריצה להדפיס אות (או לא), אם מדפיס זז צעד ימינה ולא חוזר:

$$\delta:\ Q \times \underbrace{\Sigma}_{\text{ראש סרט כתיבה}} \times \underbrace{\Gamma}_{\text{ראש סרט עבודה}} \to Q \times \underbrace{\Gamma}_{\text{כתיבה בסרט עבודה}} \times \underbrace{\{\Sigma \cup \emptyset\}}_{\text{כתיבה בסרט פלט}} \times \underbrace{\{R, L\}}_{\text{פקודת תנועה לראש סרט קלט ועבודה}}$$

מקום הריצה נמדד על סרט העבודה בלבד, לכן גם דרשנו שראש סרט הפלט לא יוכל לחזור אחורה. במכונת הכרעה עם מקום תת-לינארי, אין צורך בסרט פלט. במכונת חישוב פונ' עם מקום תת-לינארי, בסוף הריצה המכונה עוברת ל-$q_{acc}$ ועל סרט הפלט רשום $f(w)$ (הפונקציה שהמכונה מחשבת).

<u>דוגמה</u>- שפה $A \in \text{Space}(\log(n)) = L$ (מרחב שפות תת-לינאריות) אם"ם קיימת מ"ט $M$ דטרמיניסטית שמכריעה את $A$ ומסתפקת במקום $O(\log(n))$, כלומר $M$ היא מכונת 2 סרטים (קלט ועבודה) ומגבלת המקום על סרט העבודה היא $O(\log(n))$.

<u>טענה</u>

$L \subseteq P$, כלומר אם $A \in L$ אז $A \in \text{P}$.

**אם** יש מ"ט דטרמיניסטית שמסתפקת במקום לוגריתמי **אז** יש מ"ט דטרמיניסטית עבור אותה השפה שמסתפקת בזמן פולינומיאלי.

הוכחה

תהי $M$ מ"ט דטרמיניסטית שמכריעה את $A$ תוך הסתפקות במקום לוגריתמי. כמה קונפיגורציות יש למכונה?

$$|Q| \times \underbrace{n}_{\text{אפשרויות למיקום ראש הקלט}} \times \underbrace{O(\log n)}_{\text{אפשרויות למיקום ראש סרט העבודה}} \times \underbrace{|\Gamma|^{O(\log(n))}}_{\text{אפשרויות לתוכן סרט העבודה}}$$

נעריך את כמות האפשרויות לתוכן סרט העבודה:

$$|\Gamma|^{O(\log(n))} = 2^{\log_2|\Gamma| \cdot O(\log(n))} = O(n^c)$$

עבור $c$ קבוע כלשהו.

סה"כ מספר הקונפיגורציות אם כך יהיה $O(n^d)$ עבור $d$ קבוע כלשהו, בעוד ש-$M$ לא חוזרת על קונפיגורציה פעמיים כי אז הייתה בלולאה אינסופית.

**מסקנה:** זמן ריצת $M$ על $w \geq$ מספר הקונפיגורציות האפשריות שחסום ב-$O(n^d)$. כלומר, זמן ריצה פולינומי, לכן $A \in \mathrm{P}$.

טענה

אם $M$ מחשבת פונקציה $f$, ו-$M$ מ"ט דטרמיניסטית שמסתפקת במקום $O(\log(n))$ אז אורך הפלט $|f(w)|$ חסום מלמעלה על ידי פולינום.

הוכחה:

בדומה להוכחה הקודמת, מספר הקונפיגורציות חסום על ידי פולינום ולכן מספר צעדי הריצה וגם מספר ההדפסות חסום על ידי פולינום.

# שבוע 12 – הרצאה

## סיבוכיות מקום תת לינארי – תזכורת

כדי לדון במחלקות מקום תת-לינארי הגדרנו מ"ט עם 3 סרטים: סרט קלט (קריאה בלבד), עבודה (קריאה/כתיבה, ועליו בלבד מודדים את המקום), פלט (כתיבה בלבד, בכל צעד אפשר להדפיס אות או לא. אם מדפיסים, הראש זז צעד ימינה ולא חוזר שמאלה). $|w| = n$, מודדים מקום כתיבה בלבד.

מחלקות: $\mathrm{L} = \mathrm{Space}(\log n)$, $\mathrm{NL} = \mathrm{NSpace}(\log n)$.

משפט Savitch:

$$\mathrm{NSpace}\big(f(n)\big) \subsetneq \mathrm{Space}\big(f^2(n)\big)$$

לכל $f(n)$ שחשיבה במקום $O\big(f(n)\big)$. הראינו לפונקציה $f(n) \geq n$. המשפט נכון גם עבור $f(n) < n$.

תהי $f(x)$ פונקציה חשיבה במקום $O(\log n)$. כלומר, קיימת מ"ט $M_f$ עם 3 סרטים (כמוגדר למעלה) כך שלכל קלט $x$ על סרט הקלט $M_f$ מסיימת במצב $q_{acc}$ עם $f(x)$ על סרט הפלט, תוך שימוש במקום $O(\log n)$ בסרט העבודה כאשר $n$ הוא אורך הקלט. תהי $g(x)$ פונקציה חשיבה גם במקום $O(\log n)$ ע"י המכונה $M_g$.

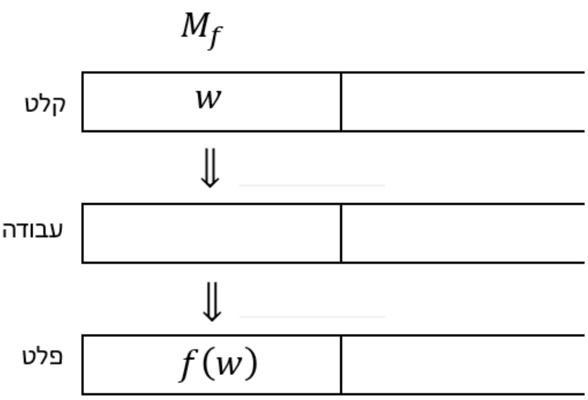
### טענה

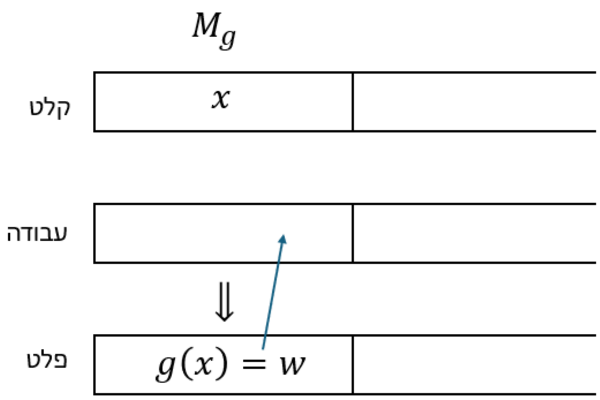
$f\big(g(x)\big)$ חשיבה במקום $O(\log n)$.

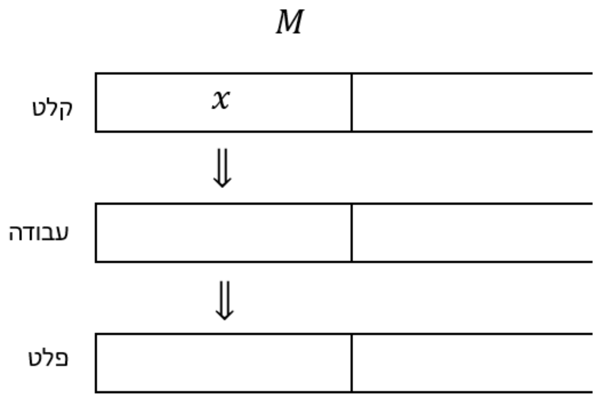
**תזכורת:** הראינו שאם $f, g$ חשיבות אז $f \circ g$ חשיבה, והראינו שאם $f, g$ חשיבות בזמן פולינומי אז $f \circ g$ חשיבה בזמן פולינומי. איך? בנינו מכונה חדשה $M$ שעבור קלט $x$ מריצה את $w = M_g(x)$, שומרת את $w$ על הסרט ואז מריצה את $M_f(w)$ וכך מקבלים את $f\big(g(x)\big)$. ברור שאם $f, g$ חשיבות הנ"ל הוכחה ש-$f \circ g$ חשיבה. בנוסף אם $f, g$ חשיבות בזמן פולינומי אז ההרכבה הנ"ל רצה בזמן פולינומי ב-$|x|$, כלומר $f \circ g$ חשיבה בזמן פולינומי.

איך אפשר להרכיב 2 מכונות של 3 סרטים כל אחת למכונת 3 סרטים שמחשבת את $f\big(g(x)\big)$?

פתרון 1 – נכון אבל בזבזני במקום: נשנה את $M_g$ כך שבמקום להדפיס לסרט הפלט, תדפיס לסרט העבודה. נשנה את $M_f$ כך שבמקום לקרוא מסרט הקלט תקרא מסרט העבודה. המכונה החדשה $M$ תפעל באופן הבא: תריץ את $M_g$ החדשה על $x$ ואז תריץ את $M_f$ החדשה (על מה שכתוב בסרט העבודה). נכונות ברורה. נותר להבין עלות מקום.




התוצאה של $M$ היא אכן $f\big(g(x)\big)$ הבעיה היא ש-$M$ עלולה להשתמש במקום שחורג ממגבלה $O(\log n)$.
הסבר: הראינו בשבוע שעבר שמ"ט דטרמיניסטית עם מגבלת מקום לוגריתמית, זמן הריצה שלהן ולכן גם אורך הפלט שלה חסומים על ידי פולינום. זה אומר ש-$|g(x)| = |w|$ עלול להיות פולינומי, ולכן המכונה $M$ עלולה להשתמש במקום פולינומי ב-$n$ במקום לוגריתמי ב-$n$.

<u>פתרון 2 – הרצה סימולטנית של $M_f$ ו-$M_g$</u>: נראה פתרון שמאפשר הרכבה כל שסה"כ מקום הריצה הוא $O(\log n)$. <u>הרעיון</u>: נחשב אותיות של $g(x)$ באופן דינמי בכל פעם מחדש. כלומר, נריץ בו זמנית את $M_f$ ואת $M_g$. הסבר מפורט יותר:

נחזיק 2 מונים, אחד שמחליף את ראש ההדפסה של $M_g$ ואחד שמחליף את ראש הקלט של $M_f$. המכונה $M$ תחשב את $f\big(g(x)\big)$ באופן הבא: תרוץ כמו $M_f$, אבל בכל צעד ריצה כדי לדעת מה האות הנוכחית מתוך $g(x)$, $M$ תריץ את $M_g$ מספר נדרש של הדפסות. כלומר: כל תנועה שמאלה של ראש קורא של $f$ נדמה על ידי הורדת מונה ב-1 שמדמה ראש סרט קלט $M_f$ וכל תנועה ימינה – הגדלה באחד של המונה. כל פקודת הדפסה של $M_g$ נדמה על ידי הגדלת מונה ראש פלט של $M_g$ ב-1. בכל צעד של $M_f$ נריץ את $M_g$ על $x$

מההתחלה (מאפסים את מונה ההדפסות). מריצים את $M_g$ עד שמונה ההדפסות שווה בערכו למונה של ראש סרט קלט $M_f$ ואז $M$ יודעת מה האות הנוכחית ש-$M_g$ מדפיסה ו-$M_f$ קוראת.

נכונות המכונה $M$ ברורה. ברור גם שזמן הריצה גדול. נרצה להשתכנע שמקום הריצה הוא $O(\log n)$. מה $M$ צריכה להחזיק על סרט העבודה?

א. מקום של סרט עבודה של $M_f$.
ב. מקום של סרט עבודה של $M_g$.
ג. מונה עבור ראש פלט $M_g$.
ד. מונה עבור ראש קלט $M_f$.

**נעריך את גודלם:**

א. $M_f$ מסתפקת במקום $O(\log|g(x)|)$ (האורך של הקלט של $M_f$, כלומר $|g(x)|$). אם כך, $M_f$ מסתפקת במקום $O(\log(n^c))$ ולכן $M_f$ מסתפקת במקום $O(\log n)$.

<u>תזכורת</u>: מכונה דטרמיניסטית שמסתפקת במקום לוגריתמי, אורך הפלט שלה חסום על ידי פולינום, כלומר אורכו $\leq n^c$ עבור $c$ קבוע כלשהו.

ב. $M_g$ מסתפקת במקום $O(\log|x|)$ לחישוב $g(x)$ כלומר $O(\log n)$.

ג. צריך מונים שיכולים לספור עד $|g(x)|$ (האורך של פלט של $M_g$, כלומר האורך של הקלט של $M_f$). הראינו כבר ב-א' ש-$|g(x)| \leq n^c$, לכן עבור מונה לאורך זה נצטרך $O(\log(n^c))$ תאים, כלומר $O(\log n)$ תאים.

ד. כנ"ל.

<u>סה"כ ל-4 הסעיפים</u>: $O(\log n)$ תאים. כלומר $M$ מכונה שמחשבת את $f\big(g(x)\big)$ תוך הסתפקות במקום $O(\log n)$. כלומר הרכבת שתי פונקציה חשיבות במקום לוגריתמי היא פונקציית חשיבה במקום לוגריתמי.

<u>הערה</u>: הראינו עבור מקום **לוגריתמי**, אפשר היה להראות גם עבור פונקציות תת לינאריות אחרות.

# המחלקות NL, L

**הגדרה – רדוקציה לוגריתמית (במקום):** עבור שפות $\boldsymbol{A, B}$ נגיד שיש רדוקציה לוגריתמית מ-$\boldsymbol{A}$ ל-$\boldsymbol{B}$ ונסמן $\boldsymbol{A \leq_L B}$ אם"ם קיימת רדוקציה $\boldsymbol{f}$ מ-$\boldsymbol{A}$ ל-$\boldsymbol{B}$ כך ש-$\boldsymbol{f}$ חשיבה במקום לוגריתמי.

הערה אם $A \leq_L B$ אז $A \leq_p B$ המכונה שמחשבת את הרדוקציה במקום לוגריתמי מספקת גם בזמן פולינומי.

שאלות:

אם $\mathrm{L} = \mathrm{NL}$ זו שאלה פתוחה. האם $\mathrm{L} = \mathrm{P}$ זו שאלה פתוחה.

האם $\mathrm{NL} = \mathrm{P}$ זו שאלה פתוחה. ראינו $\mathrm{L} \subseteq \mathrm{P}$, נראה בתרגול $\mathrm{NL} \subseteq \mathrm{P}$.

טענה – סגירות הרכבת רדוקציות

אם $A \leq_L B$ וגם $B \leq_L C$ אז $A \leq_L C$.

הוכחה: בדומה להוכחת הרכבת רדוקציות פולינומיות. נשתמש פה בסגירת הרכבת פונקציות חשיבות במקום לוגריתמי.

כלומר אם $f$ חשיבה במקום לוגריתמי היא הרדוקציה מ-$A$ ל-$B$,

וגם $g$ חשיבה במקום לוגריתמי היא הרדוקציה מ-$B$ ל-$C$,

אז $g \circ f$ חשיבה במקום לוגריתמי היא הרדוקציה מ-$A$ ל-$C$.

**הגדרה - NL-קשה:** נגיד ששפה $\boldsymbol{B}$ היא NL-קשה אם לכל $\boldsymbol{A} \in \mathrm{NL}$ מתקיים $\boldsymbol{A \leq_L B}$.

טענה

אם $B$ שפה NL-קשה וגם $B \leq_L C$ אז $C$ היא NL-קשה.

הוכחה: $B \in \text{NL-Hard}$ אז לכל $A \in \mathrm{NL}$ מתקיים $A \leq_L B$. בנוסף, $B \leq_L C$ (נתון). מתוך סגירות רדוקציות log space להרכבה נובע $A \leq_L C$ כלומר $C \in \text{NL-Hard}$.

**הגדרה - NL-שלמה:** נגיד ששפה $\boldsymbol{C}$ היא NL-שלמה אם לכל $\boldsymbol{C} \in \text{NL-Hard}$ וגם $\boldsymbol{C} \in \mathrm{NL}$.

נרצה להראות שפה שלמה ב-NL, והיא תשמש אותנו באופן דומה לשימוש של משפט קוק-לוין עבור המחלקה NP. נגדיר את השפה:

$$\mathrm{PATH} \coloneqq \{(G, s, t) \mid \text{G is a directed graph, s,t} \in \text{V, and there is a directed path from s to t in G}\}$$

לפעמים נקראת גם S-T-CONN.

טענה

$\text{PATH} \in \text{NL-Complete}$.

<u>עלינו להראות</u>:

1) $\text{PATH} \in \text{NL}$
2) $\text{PATH} \in \text{NL-Hard}$

<u>הוכחה</u>:

1) נראה מ"ט (לא דטרמיניסטית) שמסתפקת במקום $O(\log n)$ עבור השפה PATH. המכונה $M$ תפעל באופן הבא: תחזיק משתנה עבור צומת נוכחי שיאותחל ל-$s$. בכל צעד, $M$ תעבור באופן לא דטרמיניסטי לאחד השכנים של הצומת הנוכחי, ותבדוק האם הגענו ל-$t$. אם כן, $M$ תעצור ותחזיר $q_{acc}$. אם לא, תמשיך לשכן הבא.

<u>נכונות</u>: אם $(G,s,t) \in \text{PATH}$ אז קיים מסלול מכוון מ-$s$ ל-$t$ ב-$G$, לכן יש ריצה שבה $M$ מבצעת טיול מ-$s$ ל-$t$ ואז מחזירה $q_{acc}$. אם $(G,s,t) \notin \text{PATH}$ אז בכל ריצה של $M$ הטיול לא מגיע ל-$t$ ולכן $M$ לא מחזירה $q_{acc}$.

<u>מקום</u>: $M$ מחזיקה מקום על סרט העבודה עבור צומת נוכחי $O(\log n)$, ומקום עבור הצומת הבא ועוד מונה או שניים כדי לחפש קשתות בייצוג של $G$ בסרט הקלט. כל אחד ב-$O(\log n)$, סה"כ $O(\log n)$.

<u>הערות</u>:

- אפשר לשנות את $M$ כך שתמיד תעצור. למשל, להוסיף מונה אורך המסלול. מגדילים אותו באחד בכל צעד על המסלול, ואם הוא חורג ממספר הצמתים ב-$G$ עוצרים ומחזירים $q_{rej}$.
- המעבר מצומת לשכן שלו הוא בעזרת הקלט $G$. איך לעשות זאת תלוי בצורת הייצוג שדרשנו מ-$G$ (רשימת קשתות, מטריצת שכנויות וכו'). בכל מקרה, ניתן לביצוע בעזרת מונה או שניים, כלומר מקום נוסף $O(\log n)$.

2) נראה ש-$\text{PATH} \in \text{NL-Hard}$. כלומר, לכל $A \in \text{NL}$ מתקיים $A \leq_L \text{PATH}$. כלומר יש פונקציה $f$ חשיבה במקום לוגריתמי, כך שלכל $w$ מתקיים $w \in A \Leftrightarrow f(w) = (G,s,t) \in \text{PATH}$. הנתון הנוסף היחיד: קיימת מ"ט ל"ד $M$ שמסתפקת במקום לוגריתמי עבור $A$.

<u>תזכורת</u>: גרף הקונפיגורציות $G_{M,w}$ מקיים שיש מסלול מצומת הקונפיגורציה ההתחלתית $C_0$ אל צומת הקונפיגורציה המקבלת $C_{acc}$ בגרף $G_{M,w}$ אם"ם יש ריצה מקבלת של $M(w)$ אם"ם $w \in A$ (בה"כ יש קונפ' מקבלת יחידה).

לכן: עבור קלט $w$ הרדוקציה תחזיר את $(G,s,t) = \left(G_{M,w}, C_0, C_{acc}\right)$.

נכונות ברורה, נותר להראות שהרדוקציה הזו חשיבה במקום לוגריתמי.

נרצה להראות שאפשר, בהינתן $w$, לייצר את $(G_{M,w}, C_0, C_{acc})$ על ידי מ"ט דטרמיניסטית שמסתפקת במקום לוגריתמי.

$C_0$ – הקונפיגורציה ההתחלתית, כלומר סרט עבודה ריק והראשים בהתחלה, מצב פנימי $q_0$.

$C_{acc}$ – הקונפיגורציה המקבלת, כלומר סרט עבודה ריק, הראשים בהתחלה, מצב פנימי $q_{acc}$.

קל לייצר את $C_0, C_{acc}$. נותר להבין איך מדפיסים את $G_{M,w}$ ע"י מכונת log space. נבחר את הייצוג של קשתות, כלומר זוגות צמתים.

מכונת הרדוקציה $M_f$ תייצר את $G_{M,w}$ באופן הבא:

$M_f$ תחזיק זוג משתנים על סרט העבודה. כל אחד מהם מייצג קונפיגורציה. המקום הנדרש לייצוג מספר הקונפיגורציות הוא:

$$\underset{\text{מצב}}{|Q|} \times \underset{\text{מיקום ראש סרט קלט}}{n} \times \underset{\text{מיקום ראש סרט עבודה}}{O(\log n)} \times \underset{\text{תוכן סרט עבודה}}{|\Gamma|^{O(\log n)}}$$

אפשר לשמור קונפיגורציה במקום $O(\log n)$. לכל זוג הצבות לשני המשתנים המכונה תבדוק האם הם תואמים לזוג קונפיגורציות עוקבות. אם כן – תעתיק/תדפיס לסרט הפלט. בכל מקרה, גם אם לא, תקדם את המשתנים והמונים. כלומר, $M$ תעבור על כל הערכים האפשריים לשני המשתנים.

בדיקה אם זוג ערכים מתאים לזוג קונפיגורציות עוקבות – כלומר האם 2 הקונפיגורציות כמעט זהות, למעט שינויים שתואמים לפונקציה $\delta$ של $M$, ניתן לעשות במגבלת המקום.

<u>הערה</u>: כדי לשמור על קונפיגורציה אחת צריך מקום לתוכן סרט העבודה $O(\log n)$, מיקום ראש סרט עבודה $O(\log(\log n))$, מיקום ראש סרט קלט $O(\log n)$ ומצב פנימי $O(\log n)$. סה"כ $O(\log n)$.

<u>העשרה</u>: מה חסם הריצה העליון שלנו? המקום לוגריתמי, אז הזמן פולינומי.

## $\mathbf{L \subseteq NL \subseteq P \subseteq NP}$

בבל אחד מהמעברים, אנחנו לא יודעים להכריע האם יש הכלה ממש או שוויון.

$$\mathrm{L} \stackrel{?}{=} \mathrm{NL}, \mathrm{L} \stackrel{?}{=} \mathrm{P}, \mathrm{NL} \stackrel{?}{=} \mathrm{P}, \mathrm{P} \stackrel{?}{=} \mathrm{NP}$$

# שבוע 13 – הרצאה

## משפט Savitch – המשך

לכל $f(n)$ שחשיבה במקום $O\big(f(n)\big)$ מתקיים $\text{NSpace}\big(f(n)\big) \subseteq \text{Space}\big(f^2(n)\big)$. הראינו את הנ"ל לכל $f(n)$ שהיא לפחות $n \le f(n)$. נראה עכשיו שהמשפט תקף גם עבור $\log n \le f(n) \le n$.

<u>תזכורת</u>: תהי $A \in \text{NSpace}\big(f(n)\big)$ ותהי $M$ מ"ט לא דטרמיניסטית עבור $A$ שמסתפקת במקום $O\big(f(n)\big)$. עבור מילה $w$ נסתכל על גרף הקונפיגורציות $G_{M,w}$ שבו צומת לכל קונפיגורציה, קשת לכל מעבר חוקי של $\delta$ של $M$, ושתי קונפיגורציות מיוחדות $C_0$ ו-$C_{acc}$. $w \in A \iff$ קיימת ריצה מקבלת $M(w) \iff \langle G_{M,w}, C_0, C_{acc}\rangle \in \text{Path}$. כלומר, קיים מסלול מכוון מ-$C_0$ ל-$C_{acc}$ ב-$G_{M,w}$.

בנינו מכונה דטרמיניסטית $M'$ שמכריעה האם $\langle G_{M,w}, C_0, C_{acc}\rangle \in \text{Path}$ תוך הסתפקות במקום קטן יחסית: $O\big(f^2(n)\big)$. להוכחה המלאה

מה מספר הקונפיגורציות עבור מכונת NL (כאשר $f(n) = O(\log n)$). מספר הקונפיגורציות:

$$|Q| \times (n+1) \times O(\log n) \times |\Gamma|^{O(\log n)}$$

מצב | מיקום ראש סרט קלט | מיקום ראש סרט עבודה | תוכן סרט עבודה

<u>הערה 1</u>: נשים לב שבספירת הקונפיגורציות לא ספרנו את האפשרויות של סרט הקלט (כביכול $|\Sigma|^n$). מדוע? כיוון ש-$w$ הוא נתון. בהינתן קלט נתון, בין 2 קונפיגרוציות שונות, סרט הקלט זהה ולכן לא משתנה בין צמתים שונים של אותו הגרף $G_{M,w}$.

<u>הערה 2</u>: למה ה-1+ במיקום ראש סרט קלט? בגלל ה-␣ בסוף המילה.

$$\log(\#\text{conf}) = O(\log n)$$

## משפט Immerman-Szelepcsényi

<u>משפט</u>: $\text{NL} = \text{coNL}$.

כלומר, $\text{NSpace}(\log n) = \text{coNSpace}(\log n)$, כלומר $A \in \text{NL} \iff \bar{A} \in \text{NL}$.

ראינו כבר שמתקיים $\text{NPspace} = \text{PSpace}$.

<u>תזכורת</u>: $\text{NPSpace} \subseteq \text{PSpace}$ – ברור. את $\text{NPSpace} \supseteq \text{PSpace}$ מראים בעזרת משפט Savitch.

תהי $A\in \mathrm{NPSpace}$, כלומר $A\in \bigcup_{k=1}^{\infty}\mathrm{NSpace}(n^k)$. כלומר, קיים $k$ כך שמתקיים: $A\in \mathrm{NSpace}(n^k)$.
ממשפט Savitch נסיק כי $A\in \mathrm{Space}((n^k)^2)$ כלומר $A\in \mathrm{Space}(n^{2k})$ ולכן $A\in \bigcup_{k=1}^{\infty}\mathrm{Space}(n^k)=$ PSpace.

טענה:

$\mathrm{NPSpace}=\mathrm{coNPSpace}$. כלומר, $\bar{A}\in \mathrm{NPSpace} \Leftrightarrow A\in \mathrm{NPSpace}$.

הוכחה:

תהי $A\in \mathrm{NPSpace}$. נראה כי $\bar{A}\in \mathrm{NPSpace}$. הנחנו $A\in \mathrm{NPSpace}$, אבל NPSpace=PSpace ולכן $A\in$ PSpace, כלומר יש מ"ט **דטרמיניסטית** שמכריעה את $A$ תוך הסתפקות במקום פולינומי. לכן גם $\bar{A}\in$ PSpace. הסבר: אותה מכונה דטרמיניסטית מכריעה את $A$. בהחלפת $q_{acc}$ ו-$q_{rej}$ היא תהיה גם מכונת הכרעה עבור $\bar{A}$ עם אותן דרישות מקום, ולכן $\bar{A}\in \mathrm{NPSpace}$ כנדרש.

**האם יכולנו להשתמש באותה השיטה כדי להראות $\mathrm{NL}=\mathrm{coNL}$?**

לא. שימוש במשפט Savitch כדי לעבור ממ"ט לא דטרמיניסטית לדטרמיניסטית מעלה בריבוע את סיבוכיות המקום. עבור מקום פולינומי זה לא הפריע לנו, אבל עבור NL זה נותן:

$$B\in \mathrm{NL}\Longrightarrow B\in \mathrm{Space}(\log^2 n)$$

ולא ברור אם אפשר מפה לחזור ל-coNL או ל-NL. אפשר לקבל מ"ט שמכריעה את $\bar{B}$ אבל היא במקום $O(\log^2 n)$.

נתונה $A\in \mathrm{coNL}$, כלומר $\bar{A}\in \mathrm{NL}$. תהי $M$ מכונת NL עבור $\bar{A}$. **האם החלפת מצבי $q_{acc},q_{rej}$ נותנת מכונת NL עבור $A$?**

באופן כללי – לא. <u>הסבר</u>: מכונת NL עבור $\bar{A}$ מבטיחה שאם $w\in\bar{A}$ אז קיימת ריצה מקבלת, ואם $w\notin\bar{A}$ אז כל ריצה היא דוחה. היפוך המצבים $q_{acc}$ ו-$q_{rej}$ ייתן מכונה שבה אם $w\in\bar{A}$ קיימת ריצה דוחה ואם $w\notin\bar{A}$ אז כל ריצה היא מקבלת. כלומר, אם $w\notin A$ אז יש ריצה דוחה, ואם $w\in A$ אז כל ריצה מקבלת.

דוגמה קונקרטית:

$$\mathrm{Path}=\{\langle G,s,t\rangle \mid \text{There's a directed path from s to t in G}\}$$

ראינו $\mathrm{Path}\in$ NL-Complete, בפרט $\mathrm{Path}\in \mathrm{NL}$. ראינו מכונת NL עבור Path, שמתחילה מ-$s$ ובוחרת מסלול באופן ולא דטרמיניסטית ומחזירה $q_{acc}$ אם"ם המסלול מסתיים ב-$t$. החלפת מצבי $q_{acc},q_{rej}$ נותנת מכונה שלא מבטיחה נכונות עבור $\overline{\mathrm{Path}}$. בפרט, עבור $\langle G,s,t\rangle$ אם המכונה החזירה $q_{acc}$ ייתכן שזה בגלל

שאין מסלול מ-$s$ ל-$t$ ב-$G$, כלומר $\langle G,s,t\rangle \notin \overline{\text{Path}}$, אבל ייתכן שמאחר שהמכונה ניחשה מסלול אחר, ודווקא יש מסלול מ-$s$ ל-$t$ ב-$G$, כלומר המכונה הייתה אמורה להחזיר $q_{rej}$.

טענה

$\text{NL} = \text{coNL} \Leftrightarrow \overline{\text{Path}} \in \text{NL}$

הוכחה:

$\Leftarrow$ אם $\text{NL} = \text{coNL}$ אז כיוון ש-$\text{Path} \in \text{NL}$ נסיק $\overline{\text{Path}} \in \text{NL}$.

$\Rightarrow$ נראה בהכלה דו-כיוונית.

תזכורת: $\text{Path} \in \text{NL-Hard}$. תהי $A \in \text{coNL}$, אז $\bar{A} \in \text{NL}$, לכן $\bar{A} \leq_L \text{Path}$. מכאן $A \leq_L \overline{\text{Path}}$ (באמצעות אותה הרדוקציה), ונתון לנו $\overline{\text{Path}} \in \text{NL}$ אז ממשפט הרדוקציה נקבל $A \in \text{NL}$. מסקנה $\text{coNL} \subseteq \text{NL}$.

תהי $B \in \text{NL}$, לכן $\bar{B} \in \text{coNL}$. לפי ההכלה נקבל $\bar{B} \in \text{NL}$, כלומר $\text{NL} \subseteq \text{coNL}$.

לכן $\text{NL} = \text{coNL}$.

הוכחנו גרירה בשני הכיוונים ולכן $\overline{\text{Path}} \in \text{NL}$ הוא תנאי הכרחי ומספיק לכך ש-$\text{NL} = \text{coNL}$. לכן, כדי להוכיח את משפט אימרמן נרצה להראות מכונת NL עבור:

$$\overline{\text{Path}} = \{\langle G,s,t\rangle \mid \text{The directed graph G has no path from s to t}\}$$

דוגמאות לשימוש במשפט אימרמן:

דוגמה 1

$$A = \left\{\langle G,u,v\rangle \,\middle|\, \text{לא באותו רכיב קשירות חזקה בגרף המכוון } G \text{ } v,u\right\}$$

תזכורת: 2 צמתים $u,v$ באותו רכיב קשירות חזקה אם"ם יש מסלול מכוון מ-$u$ ל-$v$ וגם יש מסלול מכוון מ-$v$ ל-$u$ ב-$G$.

תזכורת: אם $C,D \in \text{NL}$ אז גם $C \cap D, C \cup D \in \text{NL}$.

צ"ל: $A \in \text{NL}$.

הוכחה: ממשפט אימרמן $\overline{\text{Path}} \in \text{NL}$. $A \in \langle G,u,v\rangle \Leftrightarrow$ אין מסלול מ-$u$ ל-$v$ או אין מסלול מ-$v$ ל-$u$ $\Leftrightarrow$ $\langle G,u,v\rangle \in \overline{\text{Path}}$ או $\langle G,v,u\rangle \in \overline{\text{Path}}$.

כלומר, איחוד של 2 שפות ב-NL ולכן $A \in \text{NL}$.

דוגמה 2

$$\text{2-Con} = \{G \mid G \text{ גרף מכוון, ויש ב } - G \text{ לפחות 2 רכיבי קשירות חזקה}\}$$

רוצים להראות $\text{2-Con} \in \text{NL}$.

הוכחה: נראה מכונת NL עבור 2-Con. $M$ תפעל באופן הבא: לכל זוג צמתים $u, v \in V(G)$ תריץ את מכונת $\overline{\text{Path}}$, על $\langle G, u, v\rangle$.

אם המכונה מחזירה $q_{acc}$ אז $M$ תענה כמוה ותעצור. אחרת $M$ תמשיך לזוג $u, v$ הבא. אם עברה על כל הזוגות אז $M$ תעצור ותחזיר $q_{rej}$.

נכונות: אם $G \in \text{2-Con}$ אז קיימים זוג צמתים $u, v$ שלא באותו רכיב קשירות חזקה. בה"כ, אין מסלול מ-$u$ ל-$v$, ואז בהרצת מכונת $\overline{\text{Path}}$ על $\langle G, u, v\rangle$ יש ריצה שתחזיר $q_{acc}$, לכן יש ריצה של $M$ שתחזיר $q_{acc}$.

אם $G \notin \text{2-Con}$ כלומר כל $G$ הוא רכיב קשירות חזקה אחד, אז לכל זוג צמתים $u, v$ כל ריצה של מכונת $\overline{\text{Path}}$ על $\langle G, u, v\rangle$ מחזירה $q_{rej}$, ולכן $M$ תמיד מחזירה $q_{rej}$.

סיבוכיות מקום: משתנה עבור $u$ ועבור $v$: $O(\log n)$. מקום עבור ריצת מכונת $\overline{\text{Path}}$ הוא $O(\log n)$. סה"כ $O(\log n)$.

## הוכחת משפט אימרמן:

צ"ל $\overline{\text{Path}} \in \text{NL}$.

רעיון כללי: נסתכל על מכונות NL שמבצעות חישובים ומבטיחות שבתום הריצה:

א. הפונקציה שמחשבים $f(n)$ רשומה בפלט, והמצב הפנימי הוא $q_{acc}$.

**או**

ב. לא בטוח מה רשום בפלט, והמצב הפנימי הוא $q_{rej}$.

**ולכל קלט יש**

ג. לפחות ריצה אחת מסתיימת ב-$q_{acc}$.

איך זה עוזר?

דוגמה: נניח שיש מכונת NL כנ"ל שנקרא לה $M'$, שעבור קלט $\langle G, s\rangle$ מחשבת את $C$, מספר הצמתים שאפשר להגיע אליהם מ-$s$ ב-$G$. בעזרת $M'$ ניתן לבנות מכונת NL בעזרת $\overline{\text{Path}}$. נריץ את $M'$ פעם אחת על $\langle G, s\rangle$ ונחשב את $C$, פעם שניה על $\langle G \setminus t, s\rangle$ ונחשב את $C'$. כלומר בפעם השנייה $M'$ מחשבת לכמה צמתים אפשר להגיע מ-$s$ כאשר מוחקים את $t$ מ-$G$.

אם $C = C'$ אז $\langle G, s, t\rangle \in \overline{\text{Path}}$. אם $C \neq C'$ אז $\langle G, s, t\rangle \in \text{Path}$. $M'$ תענה $q_{acc}$ אם"ם $C = C'$.

נסמן ב-$C_k$ את מספר הצמתים שאפשר להגיע אליהם מ-$s$ ב-$G$ על ידי $k$ צעדים או פחות.

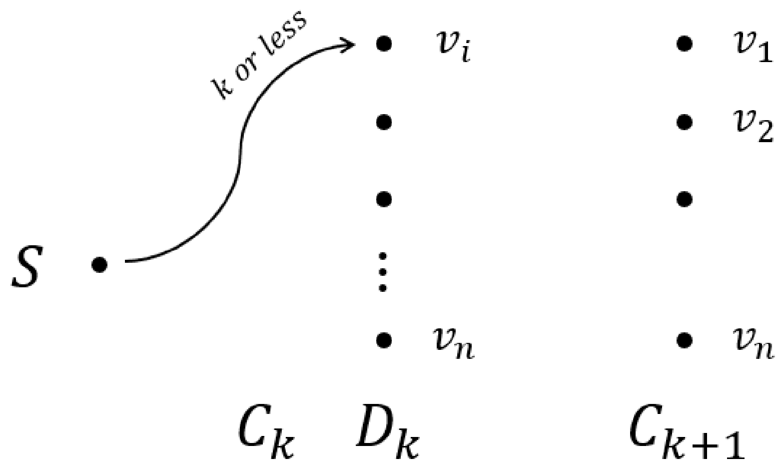
$C_0 = 1$, כי במסלול באורך $0 \geq$ אפשר להגיע ל-$s$ בלבד.

$C_1 = s$ מספר שכני $1 +$ במסלול באורך $1 \geq$ אפשר להגיע ל-$s$ או לשכן שלו.

נראה דרך איך לחשב את $C_{k+1}$ מתוך $\langle G, s, C_k \rangle$. המכונה $M''$ תפעל באופן הבא:

תתחיל מ-$C_0 = 1$ ותחשב בלולאה עד $C_n$ כאשר $n$ מספר צמתי $G$. באופן דומה נריץ את $M''$ כדי לחשב את $C'_0$ עד $C'_n$ עבור $G'$ שהוא גרף כמו $G$ אבל ללא הצומת $t$. המכונה $M$ מחשבת את $C_n$ ואת $C'_n$ ועונה $q_{acc}$ אם"ם $C_n = C'_n$.

<u>הערה</u>: המכונה $M''$ עלולה בכל שלב של הריצה לעבור למצב $q_{rej}$ אבל אם $\langle G, s, t \rangle \in \overline{\text{Path}}$ אז יש ריצה שבה $M''$ לא מחזירה $q_{rej}$ אף פעם. כלומר מחשבת נכון את הערכים $C_k$.




נתרכז במשימת החישוב של $C_{k+1}$ מתוך $C_k$. לכל צומת $v_1, \dots, v_n$ נרצה לדעת <u>בוודאות</u> האם ניתן להגיע אליו במסלול באורך $k + 1 \geq$. אם כן, נעלה את המונה $C_{k+1}$ ב-1, ואם לא אז לא. בסוף התהליך, נקבל ערך מדויק של $C_{k+1}$. איך נדע בוודאות האם אפשר להגיע ל-$v_1$ למשל, תוך $k + 1$ צעדים? אפשר, אם"ם יש צומת $v_i$ שמגיעים אליו ב-$k$ או פחות צעדים וממנו קשת אחת אל $v_1$. לכן לכל צומת $v_i \in V(G)$, מועמד להיות הלפני-אחרון במסלול מ-$s$ ל-$v_1$, ננחש מסלול מ-$s$ ל-$v_i$ באורך $k \geq$. אם הצלחנו להגיע ל-$v_i$ נעלה מונה $D_k$ ב-1, ונבדוק האם יש גם קשת $(v_i, v_1)$. אחרי שעברנו על כל ההצבות ל-$v_i$ נבדוק האם $D_k = C_k$ (כאשר $C_k$ ידוע לנו מהאיטרציה הקודמת). אם כן, סימן שעברנו על כל הצמתים במרחק $k \geq$, ואם לא סימן שפספסנו לפחות אחד. לכן, אם כן ולא הצלחנו להגיע ל-$v_1$ אז בוודאות לא ניתן להגיע אליו, אבל אם $D_k \neq C_k$ נעצור ונחזיר $q_{rej}$. באופן דומה ל-$v_1$, נעשה לכל הצמתים $v_2, \dots, v_n$ כאשר את $D_k$ מאפסים בכל פעם.

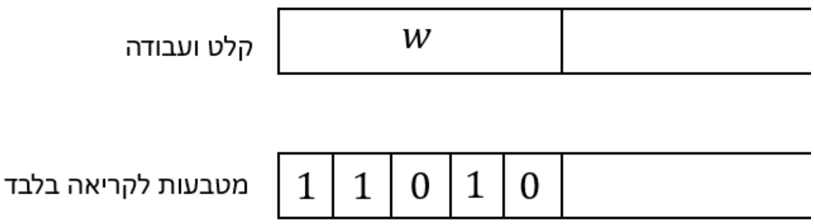
מקום: עבור $D_k, C_k, C_{k+1}$ נדרש $O(\log n)$. עבור ניחוש מסלול מ-$s$ נדרש $O(\log n)$. עבור ניחוש מסלול מ-$s$ $O(\log n)$, כי מנחשים מסלולים צעד אחרי צעד וסופרים כמה צעדים היו. עבור $C, C'$ נדרש $O(\log n)$. סה"כ $O(\log n)$.

נכונות: לכל $k$, אם לא החזרנו $q_{rej}$, מובטח ש-$C_{k+1}$ חושב נכון, מתוך $C_k$. ולכן, גם $C$ וגם $C'$ מחושבים נכון (בריצה שבה לא מחזירים $q_{rej}$) כלומר $M$ עונה נכון על $\overline{\text{Path}}$.

# שבוע 14 – הרצאה

## מחלקות סיבוכיות עם אקראיות

נרצה להרחיב את מודל מכונת טיורינג כך שיאפשר אלגוריתמים אקראיים שבהם זמן הריצה ותוצאת הריצה יכולות להיות תלויות במטבעות אקראיים. נוסיף למ"ט סרט של מטבעות אקראיים בתחילת הריצה בסרט הזה מופיעים 1 או 0 בבל תא בהסתברות שווה ובלתי-תלויה בין התאים.




נגדיר מחלקות סיבוכיות עבור מודל מ"ט זה:

**הגדרה - ZPP:** מחלקת כל השפות L כך שיש עבור מ"ט $\boldsymbol{M}$ שמכריעה את $\boldsymbol{L}$ ורצה בתוחלת זמן פולינומי (התוחלת על פני הגרלות המטבע השונות). כלומר, $\boldsymbol{M(w) = q_{acc}}$ אם"ם $\boldsymbol{w \in L}$ ואחרת $\boldsymbol{M(w) = q_{rej}}$. זמן הריצה בתוחלת (על פני המטבעות האקראיים) חסום על ידי פולינום.

טענה

$\mathrm{P} \subseteq \mathrm{ZPP}$

הוכחה: אם $L \in \mathrm{P}$ אז יש מ"ט דטרמיניסטית פולינומית שמכריעה אותה. אפשר להוסיף לה סרט מטבעות ואז נקבל מכונה שרצה תמיד בזמן פולינומי, ובפרט בתוחלת בזמן פולינומי.

**הגדרה - RP:** מכילה את כל השפות $\boldsymbol{L}$ כך שקיימת מ"ט $\boldsymbol{M}$ שרצה בזמן פולינומי (תמיד), ומתקיים: אם $\boldsymbol{w \in L}$ אז $\boldsymbol{\mathbb{P}[M(w) = q_{acc}] \geq \frac{1}{2}}$, ואם $\boldsymbol{w \notin L}$ אז $\boldsymbol{\mathbb{P}\left[M(w) = q_{rej}\right] = 1}$.

טענה

$\mathrm{RP} \subseteq \mathrm{NP}$

הוכחה: נסתכל על מוודא פולינומי לשפה ב-NP לעומת מכונת RP. בשני המקרים:

- זמן הריצה פולינומי
- אם $L \notin w$ אז $M(w) = q_{rej}$ תמיד.
- אם $w \in L$ אז יש ריצה שעבורה $M(w) = q_{acc}$. במכונת RP הדרישה היא שחצי מהריצות לפחות מסיימות ב-$q_{acc}$.

לכן מכונת RP היא מקרה פרטי של מוודא פולינומי.

טענה

$\mathrm{P} \subseteq \mathrm{RP}$

הוכחה: בשני המקרים המכונה רצה בזמן פולינומי. בשני המקרים המכונה עונה $q_{rej}$ תמיד עבור $w \notin L$. מכונה ב-P מחזירה תמיד $q_{acc}$ לקלט $w \in L$ ובפרט מחזירה $q_{acc}$ בסיכוי $\frac{1}{2} \leq$.

**הגדרה - coRP:** המחלקה coRP מכילה כל שפה $\boldsymbol{L}$ כך שמתקיים $\bar{\boldsymbol{L}} \in \mathrm{RP}$. לחלופין: $\boldsymbol{L} \in \mathrm{coRP}$ אם"ם קיימת מ"ט $\boldsymbol{M}$ עבור $\boldsymbol{L}$ כך שמתקיים:

- $M$ רצה בזמן פולינומי (תמיד).
- לכל $w \in L$ מתקיים $M(w) = q_{acc}$ (תמיד).
- לכל $w \notin L$ מתקיים $\mathbb{P}\left[M(w) = q_{rej}\right] \geq \frac{1}{2}$.

הסבר:

$$L \in \mathrm{coRP} \Leftrightarrow \bar{L} \in \mathrm{RP} \Leftrightarrow \text{יש מ"ט RP עבור } \bar{L} \Leftrightarrow \text{יש מ"ט coRP עבור L}$$

המעבר הימני ביותר על ידי החלפת מצבים $q_{acc}, q_{rej}$.

טענה

$\mathrm{RP} \cap \mathrm{coRP} = \mathrm{ZPP}$

**הגדרה - BPP:** המחלקה BPP מכילה את כל השפות $\boldsymbol{L}$ כך שיש מ"ט $\boldsymbol{M}$ שרצה בזמן פולינומי עונה נכון בסיכוי $\frac{2}{3} \leq$ (אבל מותרת טעות דו-צדדית). כלומר: אם $\boldsymbol{w} \in \boldsymbol{L}$ אז $\mathbb{P}[\boldsymbol{M}(\boldsymbol{w}) = \boldsymbol{q}_{acc}] \geq \frac{2}{3}$ ואם $\boldsymbol{w} \notin \boldsymbol{L}$ אז $\mathbb{P}\left[\boldsymbol{M}(\boldsymbol{w}) = \boldsymbol{q}_{rej}\right] \geq \frac{2}{3}$.

טענה

$\mathrm{P} \subseteq \mathrm{BPP}$

הסבר: בשני המקרים המכונה רצה בזמן פולינומי. מכונה דטרמיניסטית פולינומיות שמכריעה את $L$ בפרט מבטיחה שהסיכוי להשיב נכון הוא תמיד $\frac{2}{3} \leq$.

טענה

$\mathrm{BPP} \subseteq \mathrm{EXP}$

הסבר: תהי $L \in \mathrm{BPP}$. תהי $M$ מכונת BPP עבור $L$. נבנה בעזרתה מכונה $M'$ שמכריעה את $L$ בזמן אקספוננציאלי. $M'$ תפעל באופן הבא:

תעבור על כל האפשרויות למטבעות אקראיים. לכל אפשרות (של מחרוזת מעל $\{0,1\}$) $M'$ תריץ את $M(w)$. $M'$ תספור כמה פעמים התקבל $q_{acc}$ ותענה לפי הרוב.

הנכונות ברורה. נותר לנתח את זמן הריצה: כל הרצה של $M(w)$ דורשת זמן ריצה $\text{Poly}(n)$. כמה ריצות יש? $2^{\ell}$ כאשר $\ell$ מספר התאים בסרט המטבעות. אבל כיוון ש-$M$ רצה בזמן פולינומי היא יכולה להספיק לכל היותר מספר פולינומי של מטבעות, כלומר $\ell \subseteq \text{Poly}(n)$. לכן סה"כ זמן ריצה של $M'$ הוא $\supseteq 2^{\text{Poly}(n)}$, כלומר חסום אקספוננציאלית ב-$n$.

## בעיה

נרצה לדון על היחס בין RP,BPP. לפני כן, נראה קודם שגודל הטעות הוא קבוע קטן כרצוננו. נסמן $\text{RP}(p)$ כמו המחלקה RP אבל עם טעות חד צדדית $\geq p$. המחלקה שהגדרנו לעיל היא בעצם $\text{RP} = \text{RP}\left(\frac{1}{2}\right)$. נבחין שעם הגדרה זו מתקיים $\text{RP} = \text{RP}\left(\frac{1}{2}\right) = \text{RP}\left(\frac{1}{4}\right) = \cdots$. כלומר $L \in \text{RP}$ אם"ם קיימת מ"ט שרצה בזמן פולינומי ומקיימת שאם $w \in L$ אז $\mathbb{P}[M(w) = q_{acc}] \geq \frac{3}{4}$ ואם $w \notin L$ אז $\mathbb{P}\left[M(w) = q_{rej}\right] = 1$. ברור שמתקיים $\text{RP}\left(\frac{1}{4}\right) \supseteq \text{RP}\left(\frac{1}{2}\right)$. נותר להראות את ההכלה בכיוון השני.

תהי $L \in \text{RP}$, כלומר קיימת מ"ט $M$ שרצה בזמן פולינומי ויש לה טעות חד-צדדית בסיכוי $\geq \frac{1}{2}$. נבנה בעזרתה מ"ט $M'$ עבור $L$ שרצה בזמן פולינומי ויש לה טעות חד-צדדית בסיכוי $\geq \frac{1}{4}$.

<u>בנייה</u>: $M'$ תפעל באופן הבא – תריץ את $M$ על $w$ פעמיים, ותחזיר $q_{rej}$ אם"ם בשתי הפעמים $M(w)$ החזירה $q_{rej}$.

<u>זמן ריצה</u>: זמן הריצה של $M$ פולינומי, ולכן גם של $M'$ (פי 2).

<u>נכונות</u>: אם $w \notin L$ אז $M(w) = q_{rej}$ תמיד, ולכן $M'(w) = q_{rej}$ תמיד.

אם $w \in L$ אז $\mathbb{P}\left[M(w) = q_{rej}\right] \leq \frac{1}{2}$, לכן הסיכוי $\frac{1}{4} = \left(\frac{1}{2}\right)^2 \geq \mathbb{P}\left[M'(w) = q_{rej}\right]$.

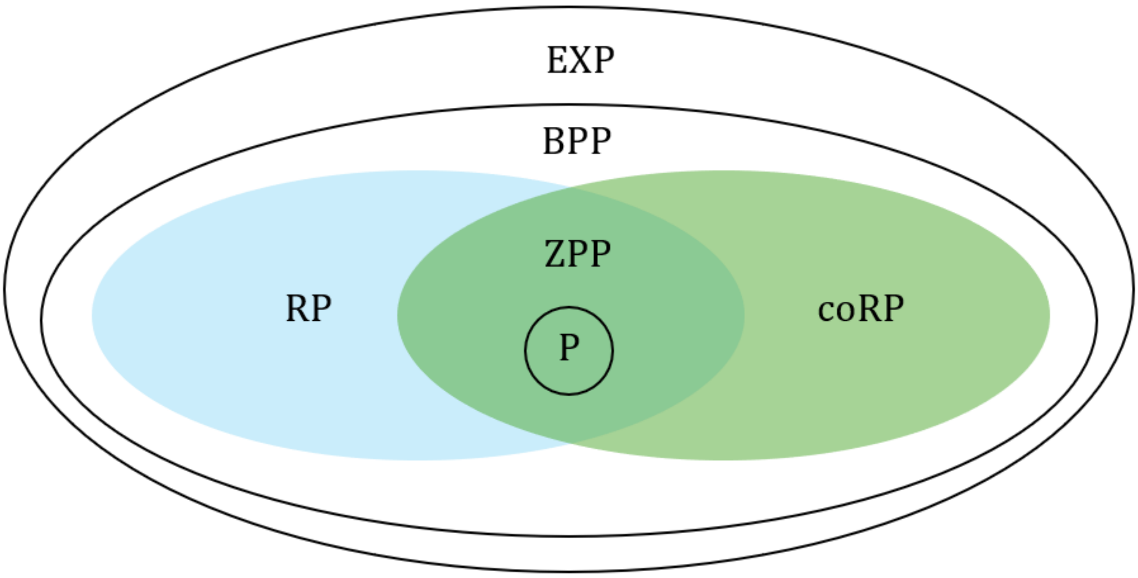
באופן דומה, יכולנו להראות $\text{RP} = \text{RP}(p)$ לכל קבוע $p$ קטן כרצוננו. כעת, נחזור ליחס ביניהן:

## טענה

$\text{RP} \subseteq \text{BPP}$

<u>הוכחה</u>: $\text{RP} = \text{RP}\left(\frac{1}{3}\right)$, כלומר $L \in \text{RP}$ אם"ם יש מ"ט פולינומיאלית עם טעות חד-צדדית $\geq \frac{1}{3}$, מקרה פרטי של מכונת BPP (שם מותר טעות דו-צדדית $\geq \frac{1}{3}$).

נרצה לקבל את התמונה המלאה:




(לא את כל ההכלות ראינו, את השאר נוכיח מטה).

טענה

$\text{coRP} \subseteq \text{BPP}$

טענת עזר: BPP סגורה למשלים, כלומר $\text{BPP} = \text{coBPP}$.

הוכחת טענת עזר: מתקבלת על ידי החלפת מצבי $q_{acc}, q_{rej}$ של מכונת BPP. נקבל מכונת BPP לשפה המשלימה.

הוכחת הטענה: $L \in \text{coRP}$ אז $\bar{L} \in \text{RP}$, כלומר $\bar{L} \in \text{BPP}$ ולכן $L \in \text{BPP}$ (מסגירות BPP למשלים).

שאלה

איפה NP משתלבת בדיאגרמה לעיל? אנחנו יודעים $\text{P} \subseteq \text{ZPP} \subseteq \text{RP} \subseteq \text{NP} \subseteq \text{EXP}$.

טענה

$\mathrm{RP} \cap \mathrm{coRP} = \mathrm{ZPP}$

<u>הוכחה</u>:

**כיוון ראשון:** נניח $L \in \mathrm{RP} \cap \mathrm{coRP}$ ונראה $L \in \mathrm{ZPP}$. כלומר, יש מכונת RP $M_1$ עבור $L$ וגם יש מכונת coRP $M_2$ עבור $L$. שתי המכונות רצות בזמן פולינומי, כל אחת עם טעות חד צדדית (שונה).

| | $\boldsymbol{M_2}$ | $\boldsymbol{M_1}$ |
|---|---|---|
| $\boldsymbol{w \in L}$ | $q_{acc}$ תמיד | $q_{acc}$ בסיכוי $\geq \frac{1}{2}$ |
| $\boldsymbol{w \notin L}$ | $q_{rej}$ בסיכוי $\geq \frac{1}{2}$ | $q_{rej}$ תמיד |

נרצה בעזת $M_1, M_2$ לבנות מ"ט $M$ שהיא מכונת ZPP עבור $L$. כלומר, <u>תמיד</u> עונה נכון. זמן הריצה של $M$ פולינומי בתוחלת (על פני המטבעות).

<u>בניה</u>: $M$ תפעל באופן הבא:

א. תריץ את $M_1(w)$. אם $M_1$ ענתה $q_{acc}$ אז $M$ תעצור ותחזיר $q_{acc}$.
ב. תריץ את $M_2(w)$. אם $M_2$ ענתה $q_{rej}$ אז $M$ תעצור ותחזיר $q_{rej}$.
ג. תחזור לשלב א'.

<u>נכונות</u>: אם בשלב כלשהו $M_1(w) = q_{acc}$ בהכרח $w \in L$ ואז התשובה של $M$ נכונה. אם בשלב כלשהו $M_2(w) = q_{rej}$ אז בהכרח $w \notin L$ ואז התשובה של $M$ נכונה. כלומר, אם $M$ עונה – היא עונה נכון.

<u>זמן ריצה</u>: נניח ש-$w \in L$. זמן הריצה של $M$ הוא זמן הריצה של שתי המכונות כפול מספר הפעמים ש-$M_1(w)$ החזירה $q_{rej}$. נסמן ב-$i$ את מספר הפעמים שזה קרה.

$\mathbb{P}[M(w) = q_{rej}] \leq \frac{1}{2}$ ולכן $\mathbb{P}[\#(M(w) = q_{rej}) = i] \leq \left(\frac{1}{2}\right)^i$, לכן תוחלת זמן הריצה היא לכל היותר $\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \cdot i \cdot \mathrm{Poly}(n)$ (הפולינום ב-$n$ הוא בעקבות חסם על זמן ריצת המכונות). הטור שקיבלנו הוא טור הנדסי ומתקיים $\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \cdot i \leq \Theta(i)$ ולכן זמן הריצה הוא לכל היותר $\Theta(i) \cdot \mathrm{Poly}(n) = \mathrm{Poly}(n)$ (הוצגה הוכחת האינטואיציה לכך).

**כיוון שני:** נניח $L \in \mathrm{ZPP}$ ונראה $L \in \mathrm{RP} \cap \mathrm{coRP}$.

נתחיל מלהראות $L \in \mathrm{ZPP} \Longrightarrow L \in \mathrm{RP}$. כלומר קיימת מ"ט שמכריעה את $L$ בזמן פולינומי בתוחלת. נבנה בעזרתה מ"ט $M_1$ שרצה בזמן פולינומי (תמיד) ויש לה טעות חד-צדדית $\frac{1}{2} \geq$.

<u>תזכורת</u>: אי-שוויון מרקוב $\mathbb{P}\left[X > a \cdot \mathbb{E}[X]\right] \leq \frac{1}{a}$ עבור מ"מ אי-שלילי $X$.

נבנה את $M_1$ באופן הבא:

- $M_1$ תריץ את $M(w)$ עם מונה למספר צעדים ותעצור את הריצה אם עברו יותר מ-$2 \cdot \mathbb{E}[\#steps]$. אם $M$ עצרה לפני כן, אז $M_1$ תשיב כמו $M$, ואם $M$ נעצרה בכוח אז $M_1$ תשיב $q_{rej}$.

<u>זמן ריצת $M_1$:</u> פי 2 מתוחלת זמן ריצת $M$ ולכן פולינומי ב-$n$.

<u>נכונות</u>: אם $w \notin L$ אז $M_1$ תמיד עונה נכון. או שעונה כמו $M$ (שתמיד עונה נכון) או שעצרה את $M$ בכוח וענתה $q_{rej}$, שגם אז זו התשובה הנכונה.

אם $w \in L$ אז $M_1$ עונה נכון אם"ם $M(w)$ סיימה לרוץ בזמן $\leq 2 \cdot$ תוחלת. כלומר, $M$ עונה לא נכון אם"ם זמן ריצת $M(w) >$ פי 2 מהתוחלת, ולפי א"ש מרקוב זה קורה בסיכוי $\leq \frac{1}{2}$. כלומר אם $w \in L$ אז $\mathbb{P}[M'(w) = q_{acc}] \geq \frac{1}{2}$. סיימנו להראות $\mathrm{ZPP} \subseteq \mathrm{RP}$.

נראה עכשיו $L \in \mathrm{ZPP} \Longrightarrow L \in \mathrm{coRP}$. כדי לעשות זאת, נבנה בעזרת $M$ מכונת $\mathrm{coRP}$ $M_2$. הבנייה כמעט בדיוק כמו $M_1$, רק שבמקרה שעוצרים את $M$ בכוח, $M_2$ תחזיר $q_{acc}$.