מודלים חישוביים, חישוביות וסיבוכיות - סיכום הרצאות

משה קול (moshe.kol@mail.huji.ac.il)

27 ביוני 2019

תקציר

סיכום הרצאות בקורס מודלים חישוביים, חישוביות וסיבוכיות (67521) כפי שהועברו באוניברסיטה העברית בירושלים בשנת 2019 ע״י פרופ׳ אורנה קופרמן.

תוכן העניינים

1 שבוע 1 - 11.03.19

1.1 הקדמה

הקורס בנוי משלושה חלקים, כפי שעולה משמו:

1. מודלים חישוביים (אוטומטים) – מודל מתמטי פורמלי שמתאר מכונת חישוב.
2. חישוביות – עוסקת בסיווג בעיות חישוביות שניתנות לפתרון ע"י מחשב לבין אלו שלא.
3. סיבוכיות – כמה משאבים דרושים לפתור בעיות מסוימות (זמן, זיכרון וכו').

סיבוכיות על קצה המזלג נתבונן בשתי בעיות חישוביות:

- בעיית מציאת מעגל אוילר: הקלט לבעיה הוא גרף לא-מכוון והפלט הוא האם קיים בגרף מעגל אוילר (מעגל שעובר בכל הצלעות בגרף פעם אחת בלבד).
- בעיית מציאת מעגל המילטון: הקלט לבעיה הוא גרף לא-מכוון והפלט הוא האם קיים בגרף מעגל המילטון (מעגל שעובר בכל הקודקודים בגרף פעם אחת בלבד).

לבעיית מציאת מעגל אוילר יש איפיון מתמטי פשוט (בגרף לא-מכוון יש מעגל אוילר אס"ם דרגות כל הקודקודים זוגיות) ולכן אלגוריתם בזמן לינארי שפותר אותה; מנגד, לבעיית מציאת מעגל המילטון, על אף "הדימיון", לא קיים איפיון מתמטי, ואלגוריתם שפותר אותה רץ בזמן אקספוננציאלי.
דוגמה נוספת לקוחה מתחום ההצפנה.

- <u>קלט:</u> שני מספרים טבעיים ראשוניים p ו-q.

  <u>פלט:</u> $n = p \cdot q$.

  קיים אלגוריתם פולינומיאלי בגודל הייצוג של המספרים.

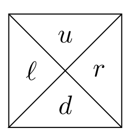
- <u>קלט:</u> $n \in \mathbb{N}$ כאשר קיימים p ו-q ראשוניים שמקיימים $p \cdot q = n$.

  <u>פלט:</u> p ו-q.

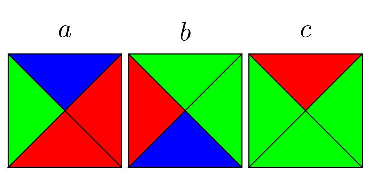
  בעיה זו היא בעיה חישובית קשה (לא ידוע אלגוריתם פולינומיאלי שפותר אותה).

לתורת הסיבוכיות יש השלכות רבות ביישומים של מדעי המחשב, ובמהלך הקורס נעמוד על חלקם.

חישוביות על קצה המזלג נתבונן בבעיה החישובית הבאה:
<u>קלט:</u> k אריחים מהצורה



כאשר u, d, r, ℓ הם צבעים.
<u>פלט:</u> להכריע האם יש ריצוף חוקי $n \times n$ לכל n בעזרת האריחים, כאשר ריצוף הוא חוקי כאשר צלעות סמוכות מסכימות על הצבע.
<u>דוגמה:</u> $k = 3$, והאריחים הם:



לכל $n \in \mathbb{N}$ ניתן לרצף שטח של $n \times n$, נדגים זאת כך:

c				
b	c			
a	b	c		
c	a	b	c	

בעיה זו נקראת ״בעיית הריצוף״, ואף על פי שהיא תמימה למראה, לא קיים אלגוריתם שפותר אותה.

1.2 מילים ושפות

הגדרה 1.1 (אלפבית). **אלפבית (א״ב)** היא קבוצה סופית לא ריקה של סימנים הנקראים **אותיות**. בדרך כלל נסמן א״ב ב-Σ.

דוגמה 1.2. א״ב נפוץ שאיתו נעבוד הוא $\Sigma = \{0, 1\}$.

הגדרה 1.3 (מילה). **מילה** מעל א״ב נתון היא סדרה סופית של אותיות מתוך א״ב זה.
מילה שלא מכילה שום אות נקראת **המילה הריקה** ומסומנת ε.
האורך של מילה w מסומן ב-$|w|$ ומוגדר בתור מספר האותיות ב-w. $|\varepsilon| = 0$.

דוגמה 1.4. הנה כמה מילים מעל הא״ב $\Sigma = \{\mathrm{a}, \mathrm{b}\}$: aabb, aaa, b.

הגדרה 1.5 (שפה פורמלית). **שפה פורמלית** L מעל א״ב Σ היא קבוצה של מילים (לאו דווקא סופית) מעל Σ.
שפת כל המילים מעל Σ מסומנת ב-Σ^*. בסימונים אלו נאמר כי שפה פורמלית L היא תת-קבוצה של Σ^*, כלומר, $L \subseteq \Sigma^*$.

דוגמה 1.6. נתבונן בא״ב $\Sigma = \{\mathrm{a}, \mathrm{b}\}$; $\Sigma^* = \{\varepsilon, \mathrm{a}, \mathrm{b}, \mathrm{aa}, \mathrm{ab}, \mathrm{ba}, \mathrm{bb}, \mathrm{aaa}, \dots\}$ היא קבוצה אינסופית שעוצמתה $\aleph_0$.

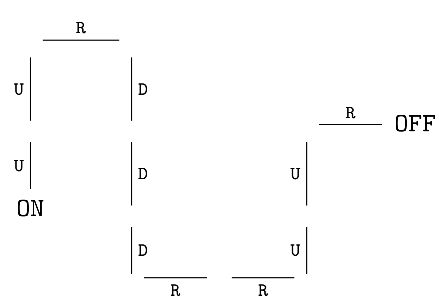
1.3 אוטומט סופי דטרמיניסטי

נפתח בדוגמה שתיתן לנו מוטיבציה להגדרת האובייקט ״אוטומט״.

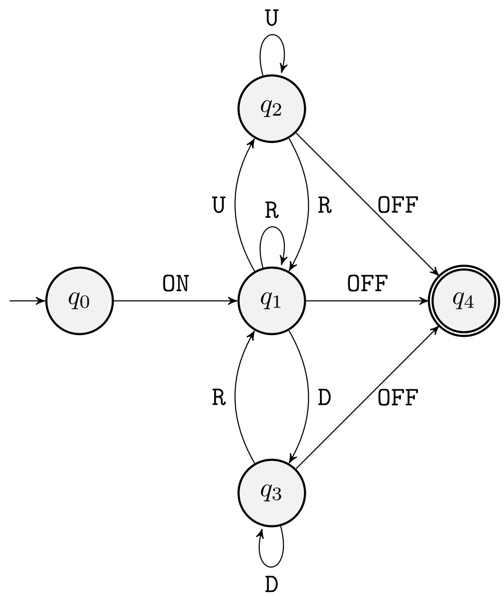
דוגמה 1.7 (עט דיגיטלי שמצייר קו-רקיע). נשווה בנפשנו עט דיגיטלי שמקבל את הפקודות ON, OFF, U, D, R, L; כאשר הפקודות U, D, R, L מציינות את הכיוונים Up, Down, Right, Left בהתאמה. נאמר שסדרת פקודות היא חוקית עבור העט אם״ם:

1. הסדרה מתחילה בפקודה ON
2. הסדרה מסתיימת בפקודה OFF
3. מציירת קו רקיע (skyline) מימין לשמאל – אסורה הפקודה L, אחרי פקודת U יכולות להתקבל רק פקודות U, R (ולא D), ואחרי פקודת D יכולות להתקבל רק פקודות D, R (ולא U).

דוגמה לסדרת פקודות חוקית: ON, U, U, R, D, D, D, R, R, U, U, R, OFF



דוגמה לסדרת פקודות לא חוקית: ON, U, R, D, L, OFF; גם הסדרה ON, U, D, OFF לא חוקית.
הבה נתאר אוטומט (סופי דטרמיניסטי) מעל הא״ב $\{$ON, OFF, U, D, R, L$\}$ שמקבל סדרת פקודות אם ורק אם היא חוקית.
את האוטומט נתאר באמצעות דיאגרמה שתכונה ״דיאגרמת מצבים״ (state diagram).



האוטומט שמתואר בדיאגרמה מורכב מחמישה מצבים q_0, q_1, q_2, q_3, q_4. המעבר על סדרת הפקודות מתחיל במצב q_0, כיוון שנכנס אליו חץ שלא מגיע מאף מצב אחר. בכל שלב האוטומט קורא פקודה מסדרת הפקודות שמוכנסת אליו כקלט, ומחליט לאיזה מצב לעבור באמצעות החצים המתאימים. אם האוטומט קורא פקודה שלא קיים עבורה חץ במצב הנוכחי, האוטומט "נתקע" (בהמשך נטפל באוטומטים שעבורם קיים חץ לכל "פקודה"). האוטומט "מקבל" סדרת פקודות אם בסוף הריצה שלו על סדרת הפקודות הוא מגיע למצב מקבל – אשר מתואר באיור בתור קודקוד עם מסגרת כפולה.

עתה להגדרה הפורמלית של אוטומט:

הגדרה 1.8 (אוטומט סופי דטרמיניסטי)**. אוטומט סופי דטרמיניסטי** (DFA) הוא חמישייה $A = \langle Q, \Sigma, q_0, \delta, F \rangle$ כאשר:

- Q היא קבוצה סופית של מצבים (states)
- Σ א"ב סופי
- $q_0 \in Q$ מצב התחלתי
- $\delta : Q \times \Sigma \to Q$ היא פונקציית מעברים (transition function)
- $F \subseteq Q$ קבוצה של מצבים מקבלים (accept states)

הערה 1.9. בדוגמה של אוטומט קו הרקיע לא לכל אות ומצב היה מצב עוקב. אנו נטפל באוטומט "שלם" במובן זה שלכל זוג של מצב ואות יש מצב עוקב (כלומר, פונקציית המצבים δ מוגדרת היטב).

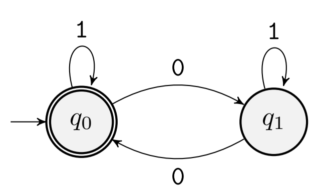
דוגמה 1.10. נתבונן באוטומט הבא:

$$A = \langle Q = \{q_0, q_1\}, \Sigma = \{0, 1\}, q_0, \delta, F = \{q_0\} \rangle$$

כאשר את פונקציית המעברים δ נתאר באמצעות טבלה:

δ	0	1
q_0	q_1	q_0
q_1	q_0	q_1

דיאגרמת המצבים של האוטומט מצוי באיור 1.1.



איור 1.1: דיאגרמת המצבים של האוטומט A

הגדרה 1.11 (ריצה של אוטומט על מילה). בהינתן מילה $w = \sigma_1\sigma_2\ldots\sigma_n$, **ריצה** של אוטומט סופי דטרמיניסטי $A = \langle Q, \Sigma, q_0, \delta, F\rangle$ על w היא סדרה של מצבים $r_0, r_1, \ldots, r_n$ כאשר:

- $r_0 = q_0$ (הריצה מתחילה במצב ההתחלתי של האוטומט)
- לכל $0 \leq i < n$ מתקיים $r_{i+1} = \delta\left(r_i, \sigma_{i+1}\right)$ (הריצה "מצייתת" לפונקציית המעברים)

דוגמה 1.12. ריצה האוטומט בדוגמה 1.10 על המילה 0010 היא הסדרה:

$$q_0, q_1, q_0, q_0, q_1$$

לפעמים נרשום $q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1$ אם נרצה לתאר את הריצה בצורה מפורטת.

הגדרה 1.13. יהי A אוטומט, w מילה ו-$r = (r_0, r_1, \ldots, r_n)$ ריצה של A על w. נאמר כי הריצה r היא **מקבלת** (או: האוטומט A **מקבל** את w) אם"ם r מסתיימת במצב מקבל, כלומר, $r_n \in F$.

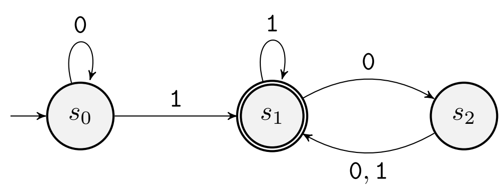
דוגמה 1.14. האוטומט בדוגמה 1.10 לא מקבל את המילה 0010 שכן ריצתו עליה מסתיימת במצב q_1 שאיננו מצב מקבל.

הגדרה 1.15 (שפה של אוטומט). יהי A אוטומט מעל הא"ב Σ. **השפה של** A (או: השפה המזוהה ע"י A) מסומנת $L\left(A\right)$ ומוגדרת ע"י:

$$L\left(A\right) \coloneqq \left\{w \in \Sigma^* : \text{the automaton } A \text{ accepts } w\right\}$$

דוגמה 1.16. מהי השפה של האוטומט מדוגמה 1.10? האוטומט מקבל את כל המילים בעלות מספר זוגי של 0-ים.

דוגמה 1.17. נתבונן באוטומט A שמתואר באמצעות דיאגרמת המצבים באיור 1.2.



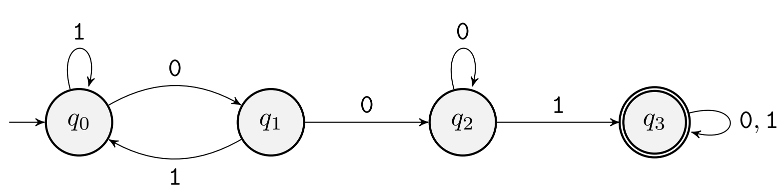
איור 1.2: דיאגרמת מצבים של האוטומט A

מהי השפה של האוטומט A? מילה $w \in \{0,1\}^*$ מתקבלת ע"י האוטומט A אם"ם יש לפחות 1 אחד, ומספר ה-0-ים אחרי ה-1 האחרון הוא זוגי.

שאלה 1.18. תאר אוטומט סופי דטרמיניסטי מעל $\Sigma = \{0,1\}$ ששפתו היא

$$\left\{w \in \Sigma^* : w \text{ contains the sequence } 001\right\}$$

פתרון. האוטומט הבא מקיים את הדרישות.



איור 1.3: תיאור גרפי של DFA מעל $\{0,1\}$ שמקבל מילה אם"ם היא מכילה את הרצף 001

המצב q_3 נקרא לעתים **בור מקבל** – אם הגענו ל-q_3, לא נוכל "לעזוב" אותו.
ננסה לתת תיאור לכל מצב באוטומט כדי לקבל אינטואיציה לבניית אוטומטים:

- q_0 – עדיין לא קראנו 001
- q_1 – קראנו 0, רוצים לקרוא 01

- q_2 – קראנו 00, רוצים לקרוא 1 (שימו לב שקריאת 0 במצב q_2 לא משנה את העובדה שקראנו 00 ; זו הסיבה לחוג העצמי ב-q_2 עבור האות 0)
- q_3 – קראנו 001 ברישא כלשהי של המילה

דוגמה 1.19 (אוטומט עם פרמטר). בהינתן $k \in \mathbb{N}$ נרצה לבנות DFA A_k מעל $\Sigma = \{0, 1\}$ ששפתו היא

$$L(A_k) = \left\{ w \in \Sigma^* : w \text{ contains the sequence } \overbrace{00\ldots0}^{k \text{ times}}1 \right\}$$

נתאר את האוטומט $A_k = \langle Q, \Sigma, q_0, \delta, F\rangle$ באופן פורמלי :
קבוצת המצבים תכיל $k+2$ מצבים $Q = \{q_0, q_1, \ldots, q_k, q_{acc}\}$. acc קיצור של accept
פונקציית המעברים δ תוגדר לכל $0 \le i < k$ באופן הבא :

$$\delta(q_i, 0) = q_{i+1}$$
$$\delta(q_i, 1) = q_0$$

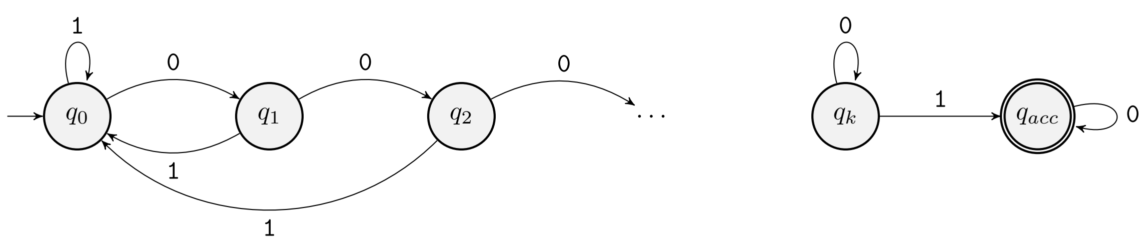
עבור q_k נגדיר :

$$\delta(q_k, 0) = q_k$$
$$\delta(q_k, 1) = q_{acc}$$

נגדיר את q_{acc} בתור בור מקבל, כלומר :

$$\delta(q_{acc}, 0) = \delta(q_{acc}, 1) = q_{acc}$$

קבוצת המצבים המקבלים הינה $F = \{q_{acc}\}$.
תיאור סכמתי של האוטומט מצוי באיור 1.4.



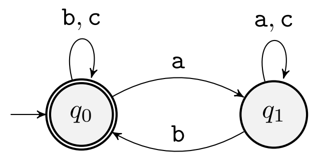
איור 1.4 : תיאור סכמתי של דיאגרמת המצבים של ה-DFA A_k

2 שבוע 1 - 13.03.19

2.1 דוגמה לאוטומט סופי דטרמיניסטי

תרגיל 2.1. תאר אוטומט סופי דטרמיניסטי מעל הא״ב $\Sigma = \{a, b, c\}$ שמקבל מילה $w \in \Sigma^*$ אס״ם אחרי כל מופע של a מופיע b. למשל, האוטומט צריך לקבל את המילים הבאות ab, aaacbccc, bccc ; ולדחות את המילים הבאות aba, ac.

פתרון. האוטומט הבא מקיים את הדרישות.



איור 2.1 : דיאגרמת מצבים של DFA שמקבל מילה אס״ם אחרי כל מופע של a יש מופע של b

2.2 פעולות על שפות

יהי Σ א״ב ו-L_1, L_2 שפות מעל Σ[1]. נגדיר את הפעולות הבאות על שפות :

1. **איחוד** –

$$L_1 \cup L_2 \coloneqq \{w \in \Sigma^* : w \in L_1 \lor w \in L_2\}$$

2. **חיתוך** –

$$L_1 \cap L_2 \coloneqq \{w \in \Sigma^* : w \in L_1 \land w \in L_2\}$$

3. **משלים** (ביחס ל-Σ^*) –

$$\bar{L}_1 \coloneqq \Sigma^* \setminus L_1$$

4. **שרשור (concatenation)** – יהיו $w_1 = a_1a_2 \dots a_n$, $w_2 = b_1b_2 \dots b_m$ מילים מעל הא״ב Σ אזי $w_1 \cdot w_2$ הוא השרשור של w_1 ו-w_2 ומוגדר כך :

$$w_1 \cdot w_2 \coloneqq a_1a_2 \dots a_nb_1b_2 \dots b_m$$

לעתים משמיטים את אופרטור השרשור $\cdot$, וכותבים w_1w_2.

שרשור של השפות L_1 ו-L_2 מוגדר ע״י :

$$L_1 \cdot L_2 \coloneqq \{w_1 \cdot w_2 : w_1 \in L_1 \land w_2 \in L_2\}$$

5. **כוכב** $*$ – אם L שפה מעל Σ, השפה L^* מציינת שרשור של 0 או יותר עותקים של L.

$$L^* = \{w_1 \cdot w_2 \cdot \dots \cdot w_k : k \in \mathbb{N} \cup \{0\}, w_i \in L \ \forall 1 \leq i \leq k\}$$

נשים לב ש- $\varepsilon \in L^*$ לכל שפה L (אפילו אם $L = \emptyset$).

בנוסף, אם $L \neq \emptyset$ וגם $L \neq \{\varepsilon\}$ אזי $|L^*| = \aleph_0$.

דוגמה 2.2. $\Sigma = \{1, 2, 3, 4\}$, $L_1 = \{1, 333\}$, $L_2 = \{22, 4444\}$.

$$L_1 \cup L_2 = \{1, 333, 22, 4444\}$$
$$L_1 \cdot L_2 = \{122, 14444, 33322, 3334444\}$$
$$L_1^* = \{\varepsilon, 1, 333, 11, 1333, 3331, 333333, 111, \dots\}$$

[1] אם L_1 היא מעל א״ב Σ_1 ו-L_2 מעל א״ב Σ_2, נגדיר $\Sigma = \Sigma_1 \cup \Sigma_2$ ונתייחס ל-L_1 ו-L_2 בתור שפות מעל הא״ב Σ

2.3 שפות רגולריות

הגדרה 2.3 (שפה רגולרית). **שפה רגולרית** מעל א״ב Σ זו שפה $\mathscr{L}$ כך שקיים אוטומט סופי דטרמיניסטי A שמקיים $L(A)=\mathscr{L}$.

לא כל שפה מעל א״ב נתון היא שפה רגולרית, הנה דוגמה:

שאלה 2.4. האם קיים DFA מעל $\Sigma=\{0,1\}$ שמקבל מילה $w\in\Sigma^*$ אס״ם מספר ה-0-ים ב-w שווה למספר ה-1-ות ב-w?

פתרון. לא. אוטומט כזה יהיה חייב לספור את מספר ה-0-ים כדי לוודא שמספר ה-1-ות שווה למספר ה-0-ים. לאוטומט סופי דטרמיניסטי יש ״זיכרון חסום״, ולא ניתן לבצע ספירה לא חסומה כנ״ל בעזרת זיכרון חסום. בהמשך נפתח כלים שיאפשרו לנו להוכיח טענות מהסגנון הזה באופן פורמלי.

סגירות השפות הרגולריות קבוצה נקראת סגורה תחת פעולה מסוימת המוגדרת עליה, כאשר הפעלת הפעולה על איברי הקבוצה נותנת איבר שנכלל אף הוא בקבוצה. למשל, קבוצת המספרים הטבעיים $\mathbb{N}$ סגורה תחת חיבור (אם $n,m\in\mathbb{N}$ אז $n+m\in\mathbb{N}$), ואינה סגורה תחת פעולת החיסור ($1\in\mathbb{N}$ ו-$2\in\mathbb{N}$ אולם $1-2\notin\mathbb{N}$).
נתעניין בפעולות סגור של השפות הרגולריות.

משפט 2.5. השפות הרגולריות סגורות תחת איחוד.

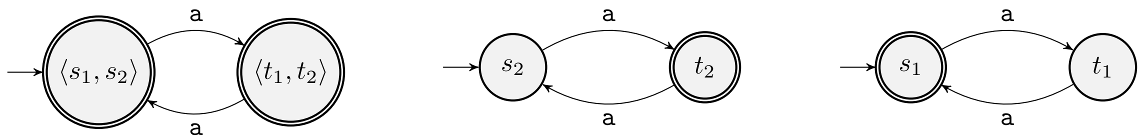
מדובר בבניה של **אוטומט המכפלה**

הוכחה. יהיו L_1 ו-L_2 שפות רגולריות מעל א״ב Σ. צריך להראות שהשפה $L_1\cup L_2$ רגולרית.
הואיל ו-L_1 שפה רגולרית, קיים אוטומט סופי דטרמיניסטי $A_1=\langle Q_1,\Sigma,q_1,\delta_1,F_1\rangle$ המזהה את השפה L_1.
הואיל ו-L_2 שפה רגולרית, קיים אוטומט סופי דטרמיניסטי $A_2=\langle Q_2,\Sigma,q_2,\delta_2,F_2\rangle$ המזהה את השפה L_2.
נבנה אוטומט סופי דטרמיניסטי $A=\langle Q,\Sigma,q_0,\delta,F\rangle$ שמזהה את השפה $L_1\cup L_2$, באופן הבא:

1. $Q=Q_1\times Q_2$, קבוצת המצבים של A תורכב מזוגות סדורים של מצב מ-Q_1 ומצב מ-Q_2.
2. $q_0=\langle q_1,q_2\rangle$, המצב ההתחלתי של A הוא הזוג הסדור $\langle q_1,q_2\rangle$ כאשר q_1 הוא המצב ההתחלתי של A_1 ו-q_2 הוא המצב ההתחלתי של A_2.
3. לכל מצב $\langle s_1,s_2\rangle\in Q$ ו-$\sigma\in\Sigma$ נגדיר את פונקציית המעברים δ כך:
$$\delta\left(\langle s_1,s_2\rangle,\sigma\right):=\langle\delta_1\left(s_1,\sigma\right),\delta_2\left(s_2,\sigma\right)\rangle$$
4. קבוצת המצבים המקבלים של A תכיל את זוגות המצבים שבהם יש מצב מקבל עבור A_1 או A_2. פורמלית:
$$\begin{aligned}F&=\left(Q_1\times F_2\right)\cup\left(F_1\times Q_2\right)\\&=\left\{\langle s_1,s_2\rangle\in Q: s_1\in F_1\vee s_2\in F_2\right\}\end{aligned}$$

נוכיח כי $L(A)=L(A_1)\cup L(A_2)$ ובכך נסיים (הרי $L(A_i)=L_i$ כי A_i מזהה את L_i, עבור $i=1,2$).
עלינו להוכיח שוויון בין קבוצות, ולכן עלינו להראות הכלה דו-כיוונית.
תהי $w\in\Sigma^*$ ונסמן $w=\sigma_1\sigma_2\ldots\sigma_n$.
ריצה של A על w היא סדרת המצבים:
$$r=\left(\langle s_1^0,s_2^0\rangle,\langle s_1^1,s_2^1\rangle,\ldots,\langle s_1^n,s_2^n\rangle\right)$$
כאשר $s_1^i\in Q_1$, $s_2^i\in Q_2$ לכל $0\le i\le n$.
לפי הבניה של פונקציית המעברים δ: $r_1=\left(s_1^0,s_1^1,\ldots,s_1^n\right)$ היא ריצה של A_1 על w, ואילו $r_2=\left(s_2^0,s_2^1,\ldots,s_2^n\right)$ היא ריצה של A_2 על w.
הריצה r היא ריצה מקבלת אס״ם $\langle s_1^n,s_2^n\rangle\in F$. לפי הגדרת F, זה קורה אס״ם $s_1^n\in F_1$ או $s_2^n\in F_2$.
כלומר, r מקבלת אס״ם r_1 מקבלת או r_2 מקבלת. לפיכך, $w\in L(A)$ אס״ם $w\in L(A_1)$ או $w\in L(A_2)$. □

דוגמה 2.6. נתבונן בא״ב $\Sigma=\{\mathrm{a}\}$ ובשפות $L_1=\{w\in\Sigma^*:|w|\equiv 0\mod 2\}$, $L_2=\{w\in\Sigma^*:|w|\equiv 1\mod 2\}$.
איור 2.2 מתאר אוטומט A_1 שמזהה את L_1, אוטומט A_2 שמזהה את L_2 ואוטומט המכפלה A שמזהה את $L=L_1\cup L_2$.



(א) האוטומט A_1

(ב) האוטומט A_2

(ג) אוטומט A (שימו לב שבעיקרון יש עוד שתי מצבים $\langle s_1, t_2 \rangle$ ו-$\langle s_2, t_1 \rangle$ אולם הם אינם ישיגים ולכן הושמטו מהאיור)

איור 2.2 : תיאור בניה של אוטומט המכפלה

3 שבוע 2 - 18.03.19

3.1 פעולות סגור של השפות הרגולריות - המשך

משפט 3.1 (השפות הרגולריות סגורות למשלים)**.** אם L שפה רגולרית מעל א״ב Σ אז גם $\bar{L}$ שפה רגולרית.

סקיצה של ההוכחה. בהינתן DFA $A = \langle Q, \Sigma, q_0, \delta, F \rangle$ שמזהה את L, ה-DFA $\bar{A} = \langle Q, \Sigma, q_0, \delta, Q \setminus F \rangle$ הוא אוטומט המזהה את $\bar{L}$. שימו לב שהשינוי היחידי בין A ו-$\bar{A}$ הוא בקבוצת המצבים המקבלים (הפעולה לפעמים נקראת: ״דואליזציה של תנאי הקבלה״). מכאן שמתקיים $w \in L(A)$ אם ורק אם $w \notin L(\bar{A})$, ונובע $L(\bar{A}) = \bar{L}$, לכן $\bar{L}$ רגולרית. (השלימו את הפרטים) □

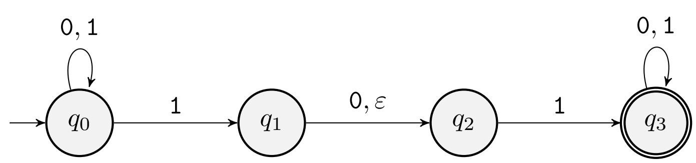
דוגמה 3.2 (סגירות השפות הרגולריות לאימפליקציה)**.** בהינתן L_1 ו-L_2 שפות רגולריות מעל א״ב Σ נגדיר את השפה $L := \{w \in \Sigma^* : w \in L_1 \to w \in L_2\}$. האם L שפה רגולרית?
התשובה חיובית, ונוכל לראות את זה בשתי דרכים:

1. ניזכר בשקילות הלוגית $(\alpha \to \beta) \equiv (\neg\alpha \vee \beta)$. מכאן נובע: $L = \bar{L}_1 \cup L_2$ ומסגירות השפות הרגולריות תחת פעולת ההשלמה והאיחוד, גם L שפה רגולרית.

2. בהינתן אוטומט סופי דטרמיניסטי $A_1 = \langle Q_1, \Sigma, q_0^1, \delta_1, F_1 \rangle$ המזהה את L_1 ואוטומט סופי דטרמיניסטי $A_2 = \langle Q_2, \Sigma, q_0^2, \delta_2, F_2 \rangle$ המזהה את L_2, נבנה את אוטומט המכפלה A (כמו בהוכחת משפט 2.5). קבוצת המצבים המקבלים של A, F_A, תוגדר באופן הבא:
$$F_A = ((Q_1 \setminus F_1) \times Q_2) \cup (Q_1 \times F_2)$$
המרכיב $((Q_1 \setminus F_1) \times Q_2)$ מתייחס לכך שהמילה לא התקבלה ע״י A_1.
המרכיב $(Q_1 \times F_2)$ מתייחס לכך שהמילה התקבלה ע״י A_2.

מה לגבי סגירות השפות הרגולריות לשרשור? מסתבר שהשפות הרגולריות סגורות לפעולת השרשור, אולם בשביל להוכיח את הטענה נמשיך ונפתח את התיאוריה של אוטומטים סופיים, ונכניס את הרעיון של ״אי-דטרמיניזם״.

3.2 אוטומט סופי לא-דטרמיניסטי (NFA)

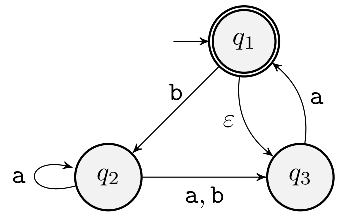


איור 3.1: אוטומט סופי לא-דטרמיניסטי שמקבל מילים מעל $\{0,1\}$ שמכילים את הרצף 101 או 11

איור 3.1 מתאר NFA. מיד נבחין במאפיינים הבאים של אוטומט סופי לא-דטרמיניסטי:

- מכל מצב יכולים לצאת כמה חצים עבור אותה אות. למשל, מעבר מ-q_0 עם האות 1 יכול להסתיים ב-q_0 או ב-q_1. זהו הרעיון של ״אי-דטרמיניזם״, בשונה מאוטומט סופי דטרמיניסטי (DFA).
- ייתכן שממצב כלשהו לא ייצא חץ עבור אות מסוימת. במקרה כזה האוטומט ״נתקע״. למשל, מהמצב q_1 לא יוצא חץ עבור האות 1.
- ממצב כלשהו יכול להיות ״מעבר-ε״ למצב אחר. ״מעבר-ε״ אומר שניתן לעבור ממצב אחד למשנהו מבלי לקרוא אות במילת הקלט. למשל, יש ״מעבר-ε״ מ-q_1 ל-q_2.

מהאמור לעיל נובע שעבור אותה מילת קלט יש מספר ריצות אפשריות ב-NFA. הריצות הללו יסתיימו (כנראה) במצבים שונים. אם עבור מילה מסוימת קיימת ריצה (אחת לפחות) שמסתיימת במצב מקבל, נאמר שהאוטומט מקבל את המילה.
בטרם נגדיר פורמלית אוטומט סופי לא-דטרמיניסטי, התבוננו באיור 3.2 ושכנעו את עצמיכם שהמילים ε, a ו-baba מתקבלות ע״י האוטומט, ואילו המילים b, bb, bab אינן מתקבלות ע״י האוטומט.



איור 3.2 : דוגמה לאוטומט סופי לא-דטרמיניסטי מעל הא״ב $\{\mathrm{a}, \mathrm{b}\}$

הגדרה 3.3 (אוטומט סופי לא-דטרמיניסטי). **אוטומט סופי לא-דטרמיניסטי** (NFA) הוא חמישייה $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ כאשר: Q, Σ, F מוגדרים כמו באוטומט הסופי הדטרמיניסטי, וגם:

- $Q_0 \subseteq Q$ - קבוצה של מצבים התחלתיים.
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ - פונקציית המעברים מעבירה מצב ואות לקבוצה של מצבים.

הערה 3.4. בתיאור דיאגרמת המצבים של אוטומט סופי לא-דטרמיניסטי לעתים נשמיט חץ ממצב מסוים q עבור אות מסוימת σ. פורמלית, ההגדרה היא $\delta(q, \sigma) = \emptyset$.

הגדרה 3.5. יהי $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ אוטומט סופי לא-דטרמיניסטי. **ריצה** של A על מילה $w \in \Sigma^*$ כאשר $w = w_1 w_2 \dots w_n$ היא סדרה של מצבים $r = (r_0, r_1, \dots, r_m)$, $m \geq n$, כאשר:

1. ניתן לכתוב את w בתור $y_1 y_2 \dots y_m \in (\Sigma \cup \{\varepsilon\})^*$ (כלומר, ניתן ״לדחוף״ ε לתוך w)
2. $r_0 \in Q_0$ (הריצה מתחילה במצב התחלתי)
3. לכל $0 \leq i < m$ מתקיים $r_{i+1} \in \delta(r_i, y_{i+1})$ (הריצה מתקבלת לפי δ)

נאמר שריצה r היא **ריצה מקבלת** אם״ם $r_m \in F$.

הגדרה 3.6. יהי $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ אוטומט סופי לא-דטרמיניסטי. השפה של A, $L(A)$, כוללת מילים שקיימת עליהן ריצה מקבלת:

$$L(A) := \{w \in \Sigma^* : \text{there exists accepting run of } A \text{ on } w\}$$

דוגמה 3.7. ריצה של האוטומט מאיור 3.1 על המילה 010110 ניתנת לתיאור כך:

$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{1} q_3 \xrightarrow{0} q_3$$

הואיל ו-q_3 מצב מקבל, המילה 010110 נמצאת בשפה של האוטומט. ריצה נוספת של המילה תתקבל אם נבטא את המילה כך: $0101\varepsilon10$, ואז:

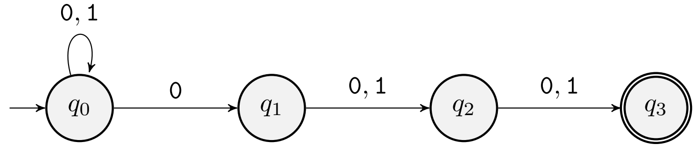
$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3$$

גם זו ריצה מקבלת.

דוגמה 3.8. נרצה אוטומט סופי לא-דטרמיניסטי מעל $\Sigma = \{0, 1\}$ ששפתו היא

$$L = \{w \in \Sigma^* : \text{there is 0 in the third position from the end}\}$$

הרעיון הוא שהאוטומט ״ינחש״ מתי מגיע ה-0 במקום השלישי מהסוף. הנה תיאור של דיאגרמת המצבים:



איור 3.3 : אוטומט סופי לא-דטרמיניסטי שמקבל את L

לשם השוואה, ניתן להוכיח שאוטומט סופי דטרמיניסטי (DFA) שמקבל את L חייב להכיל לפחות $2^3 = 8$ מצבים.

עתה נתעניין בכוח החישובי של NFA לעומת DFA. המשפט הבא עלול להפתיע:

משפט 3.9. לכל NFA A, קיים DFA שקול A' עבורו $L(A) = L(A')$.

הוכחה. נוכיח את המשפט עבור NFA שלא כולל מעברי-ε. בתרגול נראה שכל NFA שכולל מעברי-ε שקול ל-NFA שאינו כולל מעברי-ε. הבניה שנראה בהוכחה מכונה "subset construction" והוצעה לראשונה בשנת 1959 ע"י Rabin-Scott.
יהי $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ אוטומט סופי לא-דטרמיניסטי ונבנה $A' = \langle Q', \Sigma, q'_0, \delta', F'\rangle$ אוטומט סופי דטרמיניסטי באופן הבא:

- $Q' := 2^Q$.
- $q'_0 = Q_0$. שימו לב ש-$q'_0 \in Q'$.
- נגדיר $\delta'(S, \sigma) := \bigcup_{s\in S} \delta(s, \sigma)$ לכל $S \in Q'$ ו-$\sigma \in \Sigma$.
- $F' = \{S \in Q' : S \cap F \neq \emptyset\}$.

הרעיון הוא לבנות A' באופן כזה ש- A' נמצא במצב S אחרי קריאת $w \iff A$ יכול להימצא בדיוק בכל המצבים ב-S אחרי קריאת w.
ניזכר שאת פונקציית המעברים δ' ניתן להרחיב לפונקציה $\delta'^* : Q' \times \Sigma^* \to Q'$ שפועלת על זוגות של מצב ומילה:

$$\delta'^*(q', w) = \begin{cases} w = \varepsilon : & q' \\ w = u\sigma, u \in \Sigma^*, \sigma \in \Sigma : & \delta'(\delta'^*(q', u), \sigma) \end{cases}$$

בגלל הסרבול בסימון, נתייחס ל-δ'^* בתור δ', וההבנה באיזו פונקציה משתמשים תהיה ברורה מההקשר.
לכל NFA (ללא מעברי-ε) נרחיב את פונקציית המעברים δ לקבוצות של מצבים ושל מילים, כלומר, לפונקציה $\delta^* : 2^Q \times \Sigma^* \to 2^Q$, שמוגדרת באופן הבא:

$$\forall S \in 2^Q, \quad \delta^*(S, \varepsilon) := S$$
$$\forall S \in 2^Q \forall u \in \Sigma^* \forall \sigma \in \Sigma, \quad \delta^*(S, u\sigma) := \bigcup_{s\in\delta^*(S,u)} \delta(s, \sigma)$$

קבוצת המצבים שבהם A עשוי להיות אחרי קריאת $w \in \Sigma^*$ היא $\delta^*(Q_0, w)$.

טענה. לכל מילה $w \in \Sigma^*$ מתקיים: $\delta'(q'_0, w) = \delta^*(Q_0, w)$.

הוכחה. באינדוקציה על $|w|$.
בסיס: $|w| = 0 \Leftarrow w = \varepsilon$ ומתקיים:

$$\delta'(q'_0, w) = \delta'(q'_0, \varepsilon) = q'_0$$
$$\delta^*(Q_0, w) = \delta^*(Q_0, \varepsilon) = Q_0$$

ויש שוויון $q'_0 = Q_0$ לפי הגדרת האוטומט A'.
צעד האינדוקציה: נניח נכונות הטענה עבור מילים באורך $n-1$ ונוכיח את נכונותה עבור מילים באורך $n \geq 1$.
תהי $w \in \Sigma^*$ עם $|w| = n$, ונכתוב $w = u\sigma$ כאשר $u \in \Sigma^*$ ו-$\sigma \in \Sigma$. אזי:

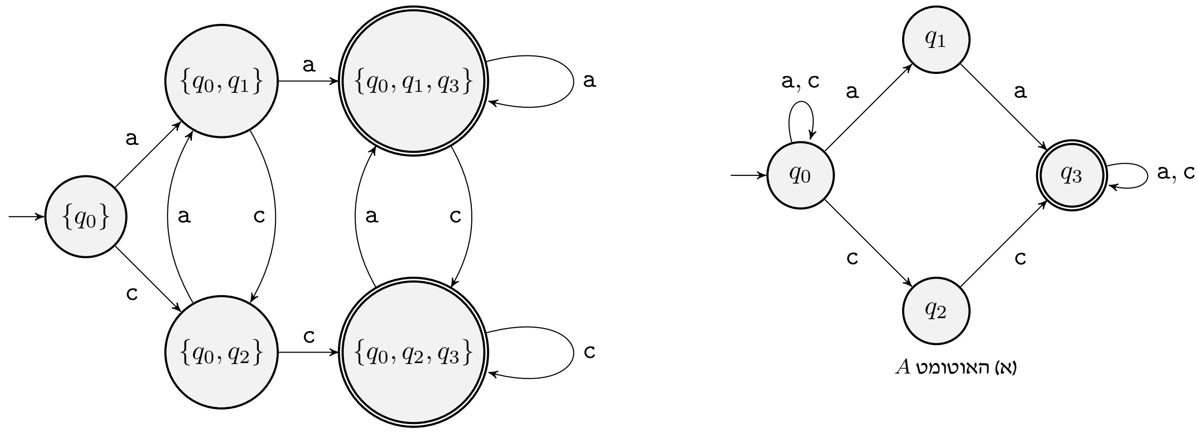
$$\begin{aligned} \delta'(q'_0, u\sigma) &= \delta'(\delta'(q'_0, u), \sigma) \\ [\text{IH}] &= \delta'(\delta^*(Q_0, u), \sigma) \\ [\text{By } \delta' \text{ definition}] &= \bigcup_{s\in\delta^*(Q_0,u)} \delta(s, \sigma) \\ [\text{By } \delta^* \text{ definition}] &= \delta^*(Q_0, u\sigma) \\ &= \delta^*(Q_0, w) \end{aligned}$$

בכך השלמנו את צעד האינדוקציה. □

להשלמת ההוכחה, נבחין כי $w \in L(A)$ אם"ם קיימת ריצה מקבלת של A על w אם"ם $\delta^*(Q_0, w) \cap F \neq \emptyset$ אם"ם $\delta^*(Q_0, w) \in F'$ אם"ם $\delta'(q'_0, w) \in F'$ אם"ם $w \in L(A')$.

□

דוגמה 3.10. באיור 3.4 אנו רואים אוטומט סופי לא-דטרמיניסטי A מעל הא"ב $\Sigma = \{\mathsf{a}, \mathsf{c}\}$ שמקבל מילה $w \in \Sigma^*$ אם"ם היא מכילה את הרצף aa או cc. לידו נמצא האוטומט A' שהוא סופי דטרמיניסטי והתקבל באמצעות ה-subset construction.



(א) האוטומט A

(ב) האוטומט A'

איור 3.4 : דוגמה להפעלת ה-subset construction

4 שבוע 2 - 20.03.19

4.1 חסם תחתון על מעבר מ-NFA ל-DFA

ראינו שניתן להעביר NFA ל-DFA בעלות של 2^n (חסם עליון) כאשר $n \in \mathbb{N}$ היא עוצמת קבוצת המצבים. נסמן זאת כך :

$$\text{NFA} \xrightarrow{2^n} \text{DFA}$$

שימו לב שהעובדה ש-2^n מופיע מעל החץ מציינת חסם עליון.
האם גם DFA $\underset{2^n}{\longrightarrow}$ NFA (חסם תחתון) נכון? מסתבר שהתשובה חיובית – אי אפשר לעשות דטרמיניזציה בפחות מ״פיצוץ אקספוננציאלי״ (exponential blow up).
ניסיון רע להוכחת חסם תחתון אקספוננציאלי : נציג שפה L שניתנת לזיהוי ע״י NFA עם 5 מצבים ו-DFA מינימלי עבורה דורש לפחות $2^5 = 32$ מצבים.
מדובר בניסיון רע, כיוון שהצגת שפה L כנ״ל לא שוללת קיום פולינום p כך ש- DFA $\xrightarrow{p(n)}$ NFA. (למשל, אם p הוא הפולינום $p(n) = n^2 + n + 2$, אז $p(5) = 32$.)
כדי להוכיח חסם תחתון DFA $\underset{2^n}{\longrightarrow}$ NFA, נציג משפחה של שפות $L_1, L_2, \dots \subseteq \Sigma^*$ כך שמתקיים :

1. לכל $n \geq 1$, השפה L_n ניתנת לזיהוי ע״י NFA עם $O(n)$ מצבים.
2. לכל $n \geq 1$ ה-DFA המינימלי עבור L_n דורש לפחות 2^n מצבים.

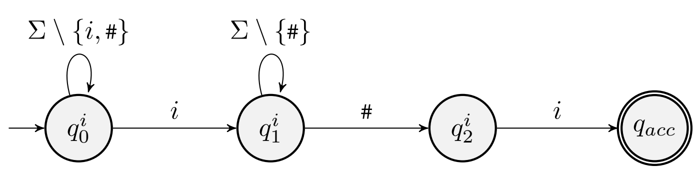
מדוע משפחה כזו של שפות סותרת קיום פולינום p כך ש- DFA $\xrightarrow{p(n)}$ NFA? לכל פולינום p, קיים $n_0 \in \mathbb{N}$ כך שמתקיים $p(n_0) < 2^{n_0}$. השפה L_{n_0} מהווה דוגמה נגדית לבניה של DFA מתוך NFA שהסיבוכיות שלה היא $p(n_0)$.
נתבונן בשפה L_n שמוגדרת מעל הא״ב Σ_n באופן הבא :

$$\Sigma_n = \{1, 2, \dots, n, \#\}$$
$$L_n = \{\sigma_1 \sigma_2 \dots \sigma_k \# \sigma_{k+1} : k \geq 1, \forall i \in [k+1] \ \ \sigma_i \in [n], \sigma_{k+1} \in \{\sigma_1, \dots, \sigma_k\}\}$$

למשל, המילה 1332#1 בשפה L_3. המילה 1332#4 לא נמצא ב-L_n לאף n. המילה ##124 גם לא חוקית.

טענה 4.1. קיים NFA שמזהה את L_n עם $O(n)$ מצבים, לכל $n \in \mathbb{N}$.

הוכחה. הרעיון הוא שה-NFA ״ינחש״ מי האות האחרונה – לכל $i \in [n]$ יהיה מצב התחלתי q_0^i שממנו אפשר לקבל את כל המילים ב-L_n שמסתיימות ב-i.



האוטומט A_n עבור L_n מוגדר פורמלית $A_n = \langle Q_n, \Sigma_n, Q_0^n, \delta_n, F_n \rangle$ כאשר:

- $Q_n = \left\{q_0^1, q_0^2, \ldots, q_0^n, q_1^1, q_1^2, \ldots, q_1^n, q_2^1, q_2^2, \ldots, q_2^n, q_{acc}\right\}$.
- $Q_0^n = \left\{q_0^1, q_0^2, \ldots, q_0^n\right\}$.
- δ_n מוגדרת בהתאם לדיאגרמת המצבים דלעיל.
- $F_n = \{q_{acc}\}$.

לפי הבניה, יש $3n + 1 = O(n)$ מצבים ב-NFA המוצע. □

טענה 4.2. ה-DFA המינימלי (מבחינת גודל קבוצת המצבים) שמזהה את L_n צריך לפחות 2^n מצבים.

הוכחה. נניח בשלילה שקיים DFA $A'_n = \langle Q', \Sigma_n, q_0, \delta', F' \rangle$ שמזהה את L_n, ויש ל-A'_n פחות מ-2^n מצבים.
עבור $S \subseteq \{1, \ldots, n\}$ תהי $w_S \in \Sigma_n^*$ המילה המורכבת מכל האותיות שב-S בסדר עולה. (למשל, אם $S = \{4, 1, 2\}$ אז $w_S = 124$).
לפי עיקרון שובך היונים, קיימות תתי-קבוצות $S_1, S_2 \subseteq \{1, 2, \ldots, n\}$ כך ש-$S_1 \neq S_2$ וגם $\delta'(q_0, w_{S_1}) = \delta'(q_0, w_{S_2})$ (משום שיש יותר תתי קבוצות של $\{1, \ldots, n\}$ מאשר מצבים ב-A'_n).
נסמן $q := \delta'(q_0, w_{S_1})$.
בה״כ, היות ש- $S_1 \neq S_2$, קיים $i \in S_1$ כך ש- $i \notin S_2$. נתבונן במצב $q' := \delta(q, \#i)$.
אם $q' \in F$ אז יש ריצה מקבלת של A'_n על $w_{S_2}\#i$ למרות שאיננה בשפה L_n.
אם $q' \notin F$ אז הריצה היחידה (כאן זה חשוב שהאוטומט דטרמיניסטי!) של A_n על $w_{S_1}\#i$ לא מקבלת, למרות שהמילה נמצאת בשפה L_n.
מכאן נסיק, A_n אינו מזהה את L_n, כלומר $L(A_n) \neq L_n$, סתירה. □

4.2 ביטויים רגולריים

הגדרה 4.3 (ביטוי רגולרי). **ביטוי רגולרי (ב״ר)** מעל א״ב Σ מוגדר באינדוקציה:

- $\emptyset$, ε ו- σ הם ב״ר, כאשר $\sigma \in \Sigma$.
- אם r_1 ו-r_2 ב״ר הרי גם $(r_1 \cup r_2)$ ב״ר. (מסומן גם: $(r_1 + r_2)$)
- אם r_1 ו-r_2 ב״ר הרי גם $(r_1 \cdot r_2)$ ב״ר.
- אם r ב״ר הרי גם (r^*) ב״ר.

ביטוי רגולרי r מעל א״ב Σ מגדיר שפה $L(r)$ באופן הבא:

- $L(\emptyset) := \emptyset$, $L(\varepsilon) := \{\varepsilon\}$, $L(\sigma) := \{\sigma\}$ כאשר $\sigma \in \Sigma$.
- אם r_1 ו-r_2 ביטויים רגולריים אז השפה של $r_1 \cup r_2$ מוגדרת ע״י $L(r_1 \cup r_2) := L(r_1) \cup L(r_2)$.
- אם r_1 ו-r_2 ביטויים רגולריים אז השפה של $r_1 \cdot r_2$ מוגדרת ע״י $L(r_1 \cdot r_2) := L(r_1) \cdot L(r_2)$.
- אם r ביטוי רגולרי אז השפה של r^* מוגדרת ע״י $L(r^*) := L(r)^*$.

הערה 4.4. לעיתים קרובות נשמיט סוגריים בביטויים רגולרים מורכבים. כדי לשמור על חד-משמעיות במשמעות ביטוי רגולרי, נגדיר סדר קדימויות: פעולת הכוכב * קודמת לפעולת השרשור $\cdot$ שקודמת לפעולת האיחוד $\cup$.

משפט 4.5. תהי $\mathscr{L}$ שפה מעל א״ב Σ. $\mathscr{L}$ רגולרית אם״ם קיים ביטוי רגולרי r כך ש- $L(r) = \mathscr{L}$.

הכיוון ״$\Rightarrow$״ בהוכחה נובע באינדוקציה על מבנה הביטוי הרגולרי ופעולות הסגור של השפות הרגולריות. הכיוון ״$\Leftarrow$״ ידון בתרגול.

דוגמה 4.6. נציג כמה ביטויים רגולריים מעל הא״ב $\Sigma = \{0, 1\}$:

- ביטוי רגולרי לשפת כל המילים שיש בהן מופע יחיד של 1:

$$0^*10^*$$

- ביטוי רגולרי לשפת כל המילים שיש בהן לפחות 1 אחד:

$$(0 \cup 1)^* 1 (0 \cup 1)^*$$

ביטוי רגולרי שקול: $0^*1(0 \cup 1)^*$.

5 שבוע 3 - 25.03.19

5.1 שפות לא רגולריות ולמת הניפוח לשפות רגולריות

טענה 5.1. לא קיים DFA שמזהה את $L = \{0^n1^n : n \geq 0\}$.

הוכחה. נניח בשלילה שקיים אוטומט סופי דטרמיניסטי A שמזהה את L. יהי p מספר המצבים ב-A.
נתבונן במילה $w = 0^p1^p$. $w \in L$ לפי הגדרת השפה L. נתבונן בריצה מקבלת של A על w:

$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{0} \ldots \xrightarrow{0} q_p \xrightarrow{1} q_{p+1} \xrightarrow{1} q_{p+2} \xrightarrow{1} \ldots \xrightarrow{1} q_{2p}$$

באוטומט יש p מצבים בסה"כ, וברישא של הריצה $\{q_0, q_1, \ldots, q_p\}$ יש $p+1$ מצבים. לפי עיקרון שובך היונים, קיימים $0 \leq \ell < j \leq p$ כך ש- $q_\ell = q_j$. נתבונן במילים $0^{p-(j-\ell)}1^p$ ו- $0^{p+i\cdot(j-\ell)}1^p$ לכל $i \geq 2$. מילים אלו אינן נמצאות בשפה L, יחד עם זאת, האוטומט A מקבל אותן. למשל, הנה ריצה מקבלת עבור $0^{p-(j-\ell)}1^p$:

$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{0} \ldots \xrightarrow{0} q_j \xrightarrow{0} q_{\ell+1} \xrightarrow{0} \ldots \xrightarrow{0} q_p \xrightarrow{1} q_{p+1} \xrightarrow{1} q_{p+2} \xrightarrow{1} \ldots \xrightarrow{1} q_{2p}$$

שימו לב שהסרנו את המקטע $q_\ell \to \cdots \to q_{j+1}$; בנוסף, $q_{\ell+1}$ הוא אכן מצב עוקב מ-q_j כי $q_\ell = q_j$.
נסיק כי האוטומט A מקבל גם מילים שאינן בשפה, וזו מהווה סתירה להנחת השלילה. □

הרעיון המרכזי בהוכחה היה שימוש בעיקרון שובך היונים והסרה/הוספה של מקטע מסוים בריצה מקבלת. רעיון זה חוזר על עצמו שוב ושוב בהוכחות מסוג זה ונציג עתה תכונה כללית של שפות רגולריות, או במילים אחרות, תנאי הכרחי לרגולריות של שפה אשר מבוסס על האבחנה הקודמת.

משפט 5.2 (למת הניפוח לשפות רגולריות, pumping lemma)**.** אם L שפה רגולרית אז קיים מספר $p \geq 1$ (קבוע הניפוח) כך שלכל מילה $w \in \Sigma^*$, אם $|w| \geq p$ ו- $w \in L$ אזי יש חלוקה $w = xyz$ המקיימת את התנאים הבאים:

1. $|y| > 0$.
2. $|xy| \leq p$.
3. לכל $i \geq 0$, $xy^iz \in L$. (ייתכן כי $x = \varepsilon$ או $z = \varepsilon$)

הוכחה. תהי L שפה רגולרית ויהי $A = \langle Q, \Sigma, q_0, \delta, F\rangle$ אוטומט סופי דטרמיניסטי עבור L. נגדיר $p := |Q|$ ונתבונן במילה כלשהי $w = \sigma_1\sigma_2\ldots\sigma_n$ כאשר $w \in L$ וגם $n \geq p$. תהי $r = (r_0, r_1, \ldots, r_n)$ ריצה מקבלת של A על w. לפי עיקרון שובך היונים, קיימים אינדקסים $0 \leq \ell < j \leq p$ כך ש- $r_\ell = r_j$. נכתוב בקיצור:

$$r_0 \xrightarrow{\sigma_1\sigma_2\ldots\sigma_\ell} r_\ell \xrightarrow{\sigma_{\ell+1}\ldots\sigma_j} r_j \xrightarrow{\sigma_{j+1}\ldots\sigma_n} r_n$$

נגדיר חלוקה של w באופן הבא:

$$x := \sigma_1 \ldots \sigma_\ell$$
$$y := \sigma_{\ell+1} \ldots \sigma_j$$
$$z := \sigma_{j+1} \ldots \sigma_n$$

נוודא שהחלוקה שהוגדרה מקיימת את שלושת התנאים בניסוח הלמה:

1. לפי הגדרת y מתקיים $|y| = j - \ell$. היות ש- $\ell < j$ נקבל $|y| > 0$.
2. $|xy| = j$ ולפי בחירת j מתקיים $j \leq p$ ולכן $|xy| \leq p$.
3. יהי $i \geq 0$, על-מנת להוכיח שהמילה xy^iz שייכת ל-L נציג ריצה מקבלת עבורה:

$$r_0 \to \cdots \to r_\ell \to (r_{\ell+1} \to \cdots \to r_j)^i \to r_{j+1} \to \cdots \to r_n \in F$$

כאשר הסימון $(\cdot)^i$ מציין שרשור i עותקים של הריצה.

□

הערה 5.3. התוודענו לעובדה שכל שפה סופית היא שפה רגולרית. כיצד שפה סופית מקיימת את תנאי למת הניפוח? במבט ראשון, נראה שהתנאי השלישי בלמת הניפוח דורש מהשפה להכיל אינסוף מילים. בחינה מחודשת של למת הניפוח מראה שלכל שפה צריך להגדיר קבוע ניפוח, p. במקרה של שפה סופית L, נגדיר את p להיות $1+\max_{w\in L}|w|$, כלומר, גדול מאורך כל מילה בשפה L. במקרה זה, הלמה מתקיימת **באופן ריק**, שהרי לא קיימת מילה w שקיימת $w\in L$ וגם $|w|\geq p$.

דוגמה 5.4. נסמן ב-L את השפה הרגולרית של הביטוי הרגולרי $(0\cup 1)^*\cdot 0\cdot(0\cup 1)$. השפה L מורכבת מהמילים שמכילים 0 באות הלפני אחרונה. נראה ש-L מקיימת את למת הניפוח. נקבע $p=3$, וניקח מילה $w\in L$ כך ש- $|w|\geq 3$. נגדיר את החלוקה הבאה: $x=\varepsilon$, y היא האות הראשונה במילה, z היא כל שאר המילה. נראה שעבור חלוקה זאת, שלושת התנאים בלמת הניפוח מתקיימים:

1. $|y|=1>0$.
2. $|xy|=1\leq 3=p$.
3. לכל $i\geq 0$ מתקיים $xy^iz\in L$ כי z מכיל את שתי האותיות האחרונות ב-xy^iz $\leftarrow$ האות הלפני האחרונה היא 0.

היפוך (contraposition) של למת הניפוח נרצה להשתמש בלמה לטובת הוכחה ששפה מסוימת אינה רגולרית. בניסוח לא מדויק: אם לכל p שמתיימר להיות קבוע הניפוח, יש מילה "רעה" כך שלכל חלוקה שלה ל-xyz שלושת התנאים בלמת הניפוח לא מתקיימים, אז השפה הנדונה לא רגולרית. הנה הניסוח המדויק:
<u>אם</u> לכל $p\geq 1$ קיימת מילה $w\in\Sigma^*$ עם $w\in L$ וגם $|w|\geq p$, ולכל חלוקה של w ל- $w=xyz$: אם $|y|>0$ וגם $|xy|\leq p$ אז קיים $i\geq 0$ כך ש- $xy^iz\notin L$, <u>אז</u> L לא רגולרית.

דוגמה 5.5. נוכיח שהשפה $L=\{w\cdot w: w\in\Sigma^*\}$ אינה רגולרית, למשל עבור $\Sigma=\{0,1\}$.
לכל $p\geq 1$ נתבונן במילה $w=0^p10^p1$. אזי $w\in L$ ובבירור $|w|\geq p$. לכל חלוקה של w ל-xyz כך ש- $|y|>0$ וגם $|xy|\leq p$ מתקיים $y\in 0^+$ וגם $xy\in 0^+$. עבור $i=2$ המילה $xyyz$ היא מהצורה $0^{p+|y|}10^p1$ והיא לא שייכת ל-L, לכן L איננה רגולרית. $0^+:=0\cdot 0^*$

5.2 איפיון שפות רגולריות

הגדרה 5.6. בהינתן שפה $L\subseteq\Sigma^*$ נגדיר יחס $\sim_L\subseteq\Sigma^*\times\Sigma^*$ (שקול-L) לכל $x,y\in\Sigma^*$ באופן הבא:

$$x\sim_L y\iff\forall z\in\Sigma^*,\quad x\cdot z\in L\leftrightarrow y\cdot z\in L$$

הערה 5.7. אם $x\not\sim_L y$ אז קיים $z\in\Sigma^*$ כך ש- $x\cdot z\in L$ ו-$y\cdot z\notin L$ או $x\cdot z\notin L$ ו-$y\cdot z\in L$. בשפה המדוברת, נכנה מילה z כנ"ל בתור "זנב מפריד".

דוגמה 5.8. ניזכר בשפה L שמתוארת ע"י $(0\cup 1)^*\cdot 0\cdot(0\cup 1)$.

1. מתקיים $111\sim_L 11$ שהרי לכל $z\in\Sigma^*$ מתקיים:

$$11\cdot z\in L\longleftrightarrow z\in L\longleftrightarrow 111\cdot z\in L$$

2. $11\not\sim_L 10$ שהרי $z=1$ הוא זנב מפריד:

$$10\cdot 1\in L\quad\wedge\quad 11\cdot 1\notin L$$

3. $0\sim_L 00$ שהרי $z=\varepsilon$ הוא זנב מפריד:

$$00\in \mathrm{L}\quad\wedge\quad 0\notin L$$

טענה 5.9. לכל שפה $L\subseteq\Sigma^*$, היחס $\sim_L$ הוא יחס שקילות.

הוכחה. יחס שקילות הוא יחס רפלקסיבי, סימטרי וטרנזיטיבי. נוודא את שלושת התכונות הללו עבור $\sim_L$:

1. רפלקסיביות: $x\sim_L x$ שהרי לכל $z\in\Sigma^*$ מתקיים $x\cdot z\in L\leftrightarrow x\cdot z\in L$ (זוהי טאוטולוגיה).
2. סימטריות: אם $y\sim_L x$ אז לכל $z\in\Sigma^*$ מתקיים $y\cdot z\in L\leftrightarrow x\cdot z\in L$, ומסימטריות האופרטור $\leftrightarrow$ (אם"ם) אנו מקבלים $y\sim_L x$.

3. טרנזיטיביות: נניח כי $w_1 \sim_L w_2$ וגם $w_2 \sim_L w_3$ ואילו $w_1 \not\sim_L w_3$. זה אומר שקיים $z \in \Sigma^*$ כך שמתקיים (בלי הגבלת הכלליות) $w_1 \cdot z \in L$ ו- $w_3 \cdot z \notin L$. הואיל ו- $w_2 \sim_L w_3$ אנו מקבלים $w_2 \cdot z \notin L$. כיוון ש- $w_1 \sim_L w_2$ אנו מקבלים $w_1 \cdot z \notin L$, סתירה. נסיק כי אם $w_1 \sim_L w_2$ וגם $w_2 \sim_L w_3$ אז גם $w_1 \sim_L w_3$, ולכן היחס טרנזיטיבי.

□

מסקנה מן הטענה האחרונה היא שהיחס $\sim_L$ מחלק את Σ^* למחלקות שקילות.

צורת רישום 5.10. עבור $L \subseteq \Sigma^*$ ו- $w \in \Sigma^*$ נסמן את מחלקת השקילות של w ב- $[w]$, כאשר:

$$[w] \coloneqq \{w' \in \Sigma^* : w \sim_L w'\}$$

דוגמה 5.11. מחלקות השקילות של $\sim_L$ עבור $L = (0 \cup 1)^* \cdot 0 \cdot (0 \cup 1)$ הן:

$$S_1 = \varepsilon \cup 1 \cup \Sigma^* 11$$
$$S_2 = 0 \cup \Sigma^* 10$$
$$S_3 = \Sigma^* 00$$
$$S_4 = \Sigma^* 01$$

בתרשים הבא מצויים הזנבות המפרידים בין כל זוג של מחלקות שקילות (שימו לב שהזנבות המפרידים תקפים לכל מילה במחלקת השקילות):

$$\begin{tikzcd}
{[0]} \arrow[r, leftrightarrow, "0 \cup 1"] \arrow[d, leftrightarrow, "\varepsilon"'] \arrow[dr, leftrightarrow, "\varepsilon"] & {[\varepsilon]} \arrow[d, leftrightarrow, "\varepsilon"] \arrow[dl, leftrightarrow, "\varepsilon"'] \\
{[00]} \arrow[r, leftrightarrow, "0 \cup 1"] & {[01]}
\end{tikzcd}$$

דוגמה 5.12. נטען שמספר מחלקות השקילות של $\sim_L$ עבור $L = \{0^n 1^n : n \geq 0\}$ אינו סופי. עבור $i < j$ מתקיים $0^i \not\sim_L 0^j$ כי 1^i הוא זנב מפריד. מכאן נובע שמספר מחלקות השקילות של $\sim_L$ אינו סופי שהרי כל מילה 0^i נמצאת במחלקת שקילות נפרדת מכל שאר המילים 0^j.

6 שבוע 3 - 27.03.19

6.1 משפט Myhill-Nerode

משפט 6.1 (משפט Myhill-Nerode)**.** לכל שפה $L \subseteq \Sigma^*$ מתקיים: L רגולרית $\iff$ מספר מחלקות השקילות של $\sim_L$ סופי.

הוכחה. $(\Leftarrow)$: נניח כי L רגולרית ונוכיח שמספר מחלקות השקילות של $\sim_L$ סופי. יהי $A = \langle Q, \Sigma, q_0, \delta, F \rangle$ אוטומט סופי דטרמיניסטי שמזהה את L. נגדיר את היחס $\sim_A \subseteq \Sigma^* \times \Sigma^*$ (שקול-A) לכל $x, y \in \Sigma^*$ באופן הבא:

$$x \sim_A y \iff \delta(q_0, x) = \delta(q_0, y)$$

כלומר $x \sim_A y$ אס"ם x ו-y מגיעים לאותו המצב ע"י האוטומט A.
נטען כי $x \sim_A y \Leftarrow x \sim_L y$ שהרי לכל $z \in \Sigma^*$ מתקיים:

$$\begin{aligned} \delta(q_0, xz) &= \delta(\delta(q_0, x), z) \\ &= \delta(\delta(q_0, y), z) \\ &= \delta(q_0, yz) \end{aligned}$$

מכאן נובע שמספר מחלקות השקילות של $\sim_L$ הוא לכל היותר מספר מחלקות השקילות של $\sim_A$. מספר מחלקות השקילות של $\sim_A$ הוא לכל היותר $|Q|$ (כל המצבים הישיגים מן המצב ההתחלתי q_0), שהוא מספר סופי. לכן מספר מחלקות השקילות של $\sim_L$ סופי, כנדרש.
$(\Rightarrow)$: נתון שמספר מחלקות השקילות של $\sim_L$ סופי ועלינו להראות שהשפה L רגולרית. לשם כך נבנה אוטומט סופי דטרמיניסטי $A = \langle Q, \Sigma, q_0, \delta, F \rangle$ שעבורו מתקיים $L(A) = L$. נגדיר:

- Q – המצבים באוטומט יהיו מחלקות השקילות של $\sim_L$. שימו לב שמספר המצבים סופי.
- המצב ההתחלתי q_0 יוגדר ע"י $q_0 := [\varepsilon]$ (מחלקת השקילות של ε).
- פונקציית המעברים δ תוגדר לכל מצב $[w]$ ואות $\sigma \in \Sigma$ באופן הבא:

$$\delta([w], \sigma) = [w \cdot \sigma]$$

נשים לב שההגדרה של δ לא תלויה בבחירה של נציג מחלקת השקילות: אם $x \sim_L y$ אז לכל $\sigma \in \Sigma$ מתקיים $x\sigma \sim_L y\sigma$. לפיכך, אם $[x] = [y]$ (אפילו אם $x \neq y$) אז $\delta([x], \sigma) = \delta([y], \sigma)$.

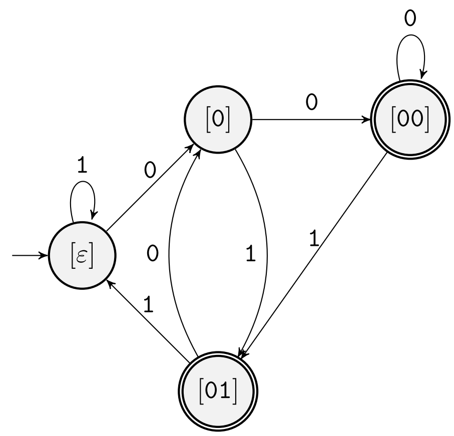
- קבוצת המצבים המקבלים מוגדרת ע"י $F := \{[w] : w \in L\}$. שימו לב שגם ההגדרה של F לא תלויה בבחירת הנציג – אם $x \sim_L y$ אז $[x] \in F \leftrightarrow [y] \in F$.

נטען כי $L(A) = L$. טענה זו נובעת בקלות מן הטענה שלכל $w \in \Sigma^*$ מתקיים $\delta(q_0, w) = [w]$.
ההוכחה מתבצעת באינדוקציה על $|w|$ – אורך המילה. בסיס האינדוקציה $|w| = 0$ מתקיים כי $\delta(q_0, \varepsilon) = q_0 = [\varepsilon]$ לפי הגדרת q_0. בצעד האינדוקציה, נניח שהטענה מתקיימת עבור $|w| = n - 1$ ונוכיח עבור $|w| = n$. נכתוב $w = w'\sigma$ כאשר $w' \in \Sigma^*$ ו-$\sigma \in \Sigma$:

$$\begin{aligned} \delta(q_0, w) &= \delta(q_0, w'\sigma) \\ &= \delta(\delta(q_0, w'), \sigma) \\ &= \delta([w'], \sigma) && \text{הנחת האינדוקציה} \\ &= [w' \cdot \sigma] && \delta \text{ לפי הגדרת} \\ &= [w] \end{aligned}$$

בכך סיימנו את צעד האינדוקציה. עתה מתקיים: $w \in L(A)$ אס"ם $\delta(q_0, w) \in F$ אס"ם $[w] \in F$ אס"ם $w \in L$. לכן, $L(A) = L$. בכך הגדרנו אוטומט סופי דטרמיניסטי שמזהה את L ולכן L רגולרית. $\square$

דוגמה 6.2. היזכרו בדוגמה 5.11 בה הצגנו את השפה $L = (0 \cup 1)^* \cdot 0 \cdot (0 \cup 1)$ ואת מחלקות השקילות שלה. באיור 6.1 מוצג אוטומט סופי דטרמיניסטי שמזהה את L שעולה מן הבניה שמוצגת במשפט Myhill-Nerode.



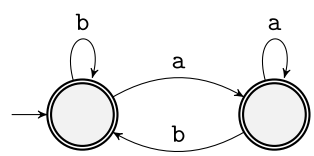
איור 6.1: אוטומט סופי דטרמיניסטי עבור השפה L

7 שבוע 4 - 01.04.19

7.1 מינימיזציה של אוטומטים סופיים דטרמיניסטיים

מהוכחת הכיוון הראשון של משפט 6.1 נובע שלא ייתכן DFA עבור שפה מסוימת L עם פחות מצבים ממספר מחלקות השקילות של $\sim_L$ (שהרי אז היו קיימות שתי מילים x ו-y במחלקות שקילות שונות שמגיעות לאותו המצב באוטומט). לפיכך, האוטומט שהגדרנו בכיוון השני של משפט 6.1 הוא מינימלי (ביחס למספר המצבים). כעת נרצה להציג אלגוריתם שבהינתן אוטומט סופי דטרמיניסטי עבור השפה L מחזיר אוטומט סופי דטרמיניסטי שקול (כלומר, מזהה את L) עם מספר מצבים מינימלי השווה למספר מחלקות השקילות של $\sim_L$.

באיור 7.1 אנו רואים דוגמה ל-DFA שאינו מינימלי. ליחס $\sim_A$, שהוגדר בהוכחת משפט 6.1, יש שתי מחלקות שקילות, בעוד שליחס $\sim_{L(A)}$ יש רק מחלקת שקילות אחת.



איור 7.1: דוגמה ל-DFA A שאינו מינימלי עבור השפה $(\mathtt{a}\cup\mathtt{b})^*$

לצורך האלגוריתם, נגדיר יחס שקילות בין מצבי אוטומט סופי דטרמיניסטי. נגדיר את היחס $\equiv_A^i$ בצורה אינדוקטיבית/חישובית ולאחר מכן נוכיח איפיון שקול שלו.

הגדרה 7.1. יהי $A=\langle Q,\Sigma,q_0,\delta,F\rangle$ אוטומט סופי דטרמיניסטי. לכל $i\geq 0$ נגדיר יחס $\equiv_A^i\subseteq Q\times Q$ לכל $s,s'\in Q$ באופן הבא:

$$
\begin{aligned}
s\equiv_A^0 s' &\iff (s\in F\leftrightarrow s'\in F)\\
s\equiv_A^i s' &\iff \left(s\equiv_A^{i-1} s' \ \wedge\ \forall\sigma\in\Sigma\ \ \delta(s,\sigma)\equiv_A^{i-1}\delta(s',\sigma)\right)
\end{aligned}
$$

הערה 7.2.

- היחס $\equiv_A^i$ מוגדר על מצבים של אוטומט סופי דטרמיניסטי בעוד שהיחסים $\sim_A$ ו-$\sim_L$ הוגדרו על מילים.
- היחס $\equiv_A^i$ הוא יחס שקילות לכל $i\geq 0$ (וודאו!).
- כפי שברור מן ההגדרה, מחלקות השקילות של $\equiv_A^0$ הן F ו-$Q\setminus F$. ההגדרה האינדוקטיבית של $\equiv_A^i$ מתארת למעשה אלגוריתם (יעיל) למציאת מחלקות השקילות של $\equiv_A^i$ מתוך מחלקות השקילות של $\equiv_A^{i-1}$.

למה 7.3. יהי $A=\langle Q,\Sigma,q_0,\delta,F\rangle$ אוטומט סופי דטרמיניסטי. לכל $i\geq 0$ ולכל $s,s'\in Q$ מתקיים:

$$s\equiv_A^i s' \iff \forall w\in\Sigma^*\text{ such that }|w|\leq i\text{ it holds that }\delta(s,w)\in F\leftrightarrow\delta(s',w)\in F$$

כלומר, המצבים s ו-s' מסכימים עבור מילים באורך קטן-שווה מ-i.

הוכחה. נוכיח את הטענה באינדוקציה על i.
מקרה הבסיס: $i = 0$. במקרה זה $w = \varepsilon$ ומתקיים:

$$\delta(s,\varepsilon) = s$$
$$\delta(s',\varepsilon) = s'$$

מכאן נובע:

$$(\delta(s,\varepsilon) \in F \leftrightarrow \delta(s',\varepsilon) \in F) \iff (s \in F \leftrightarrow s' \in F) \iff s \equiv_A^0 s'$$

צעד האינדוקציה: נניח את נכונות הטענה עבור $i-1$ ונוכיח את נכונותה עבור i.
לכל מילה $w \in \Sigma^*$ שמקיימת $|w| \leq i$ מתקיים $(\delta(s,w) \in F \leftrightarrow \delta(s',w) \in F)$ אם ורק אם לכל מילה $w' \in \Sigma^*$ כך ש- $|w'| \leq i-1$ מתקיים $(\delta(s,w') \in F \leftrightarrow \delta(s',w') \in F)$ וגם לכל מילה $w' \in \Sigma^*$, $|w'| \leq i-1$, ואות $\sigma \in \Sigma$ מתקיים $(\delta(s,\sigma w') \in F \leftrightarrow \delta(s',\sigma w') \in F)$, או באופן שקול, $(\delta(\delta(s,\sigma),w') \in F \leftrightarrow \delta(\delta(s',\sigma),w') \in F)$. לפי הנחת האינדוקציה, זה קורה אם ורק אם $s \equiv_A^{i-1} s'$ וגם $\delta(s,\sigma) \equiv_A^{i-1} \delta(s',\sigma)$ (שהרי $|w'| \leq i-1$). לפי הגדרת היחס $\equiv_A^i$, זה קורה אם ורק אם $s \equiv_A^i s'$, כנדרש. □

הערה 7.4. בהינתן אוטומט סופי דטרמיניסטי $A = \langle Q,\Sigma,q_0,\delta,F\rangle$ נסמן ב- A^s את האוטומט A עם מצב התחלתי s, כלומר, $A^s = \langle Q,\Sigma,s,\delta,F\rangle$ עבור $s \in Q$. המצבים s ו-s' מסכימים על כל המילים (באורך כלשהו) אס״ם $L(A^s) = L\left(A^{s'}\right)$.
בנוסף, $s \equiv_A^i s'$ אס״ם $L(A^s) \cap \Sigma^{\leq i} = L\left(A^{s'}\right) \cap \Sigma^{\leq i}$. זוהי דרך שקולה לבטא את אותו תנאי שרשום בלמה 7.3.

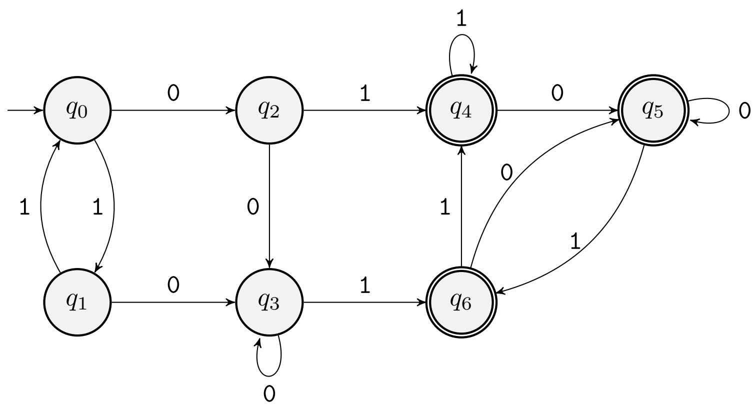
סדרת היחסים שהוגדרה $\equiv_A^0, \equiv_A^1, \equiv_A^2, \ldots$ מגיעה **לנקודת שבת**. כלומר, אנו טוענים שקיים אינדקס מינימלי i^* כך שהיחס $\equiv_A^{i^*}$ זהה ליחס $\equiv_A^{i^*+1}$. כדי לראות זאת, נתבונן במחלקות השקילות של $\equiv_A^i$. ליחס $\equiv_A^0$ יש שתי מחלקות שקילות האחת F (קבוצת המצבים המקבלים) והשניה $Q \setminus F$ (קבוצת המצבים שאינם מקבלים). לפי הגדרת $\equiv_A^i$, ליחס $\equiv_A^i$ יש יותר מחלקות שקילות מאשר ליחס $\equiv_A^{i-1}$ (כי אם $s \equiv_A^i s'$ אז $s \equiv_A^{i-1} s'$). מספר מחלקות השקילות הגדול ביותר שיכול להיות הוא $|Q|$ (כל מצב נמצא במחלקת שקילות משל עצמו). לכן, $i^* \leq |Q|$, ולכן מובטח שנגיע לנקודת שבת אחרי לכל היותר $|Q|$ איטרציות.
הטענה המרכזית היא שמספר מחלקות השקילות של $\equiv_A^{i^*}$ שווה למספר מחלקות השקילות של היחס $\sim_{L(A)}$. יתרה מזאת, האוטומט המינימלי הוא יחיד, עד כדי שינוי שמות המצבים. (לא הובאה הוכחה לטענה הזאת ונאמר כי היא תינתן בתרגול.)

הגדרת DFA מינימלי בהינתן אוטומט סופי דטרמיניסטי $A = \langle Q,\Sigma,q_0,\delta,F\rangle$ אנו נגדיר אוטומט סופי דטרמיניסטי מינימלי $A' = \langle Q',\Sigma,q_0',\delta',F'\rangle$ באופן הבא:

- $Q' := \{[q] : q \in Q\}$, כאשר הסימון $[\cdot]$ מתייחס למחלקות השקילות של היחס $\equiv_A^{i^*}$, כאשר i^* הוא האינדקס המינימלי שמקיים שהיחס $\equiv_A^{i^*}$ זהה ליחס $\equiv_A^{i^*+1}$.
- $q_0' := [q_0]$.
- פונקציית המעברים תוגדר ע״י $\delta'([q],\sigma) := [\delta(q,\sigma)]$.
- קבוצת המצבים המקבלים תוגדר ע״י $F' := \{[q] : q \in F\}$.

בהגדרת A' צריך להצדיק מדוע אין השפעה של בחירת הנציג של מחלקת השקילות – ההצדקה דומה להגדרת האוטומט במשפט 6.1.

דוגמה 7.5. נתבונן באוטומט הסופי דטרמיניסטי הבא:



איור 7.2: אוטומט סופי דטרמיניסטי A שאינו מינימלי

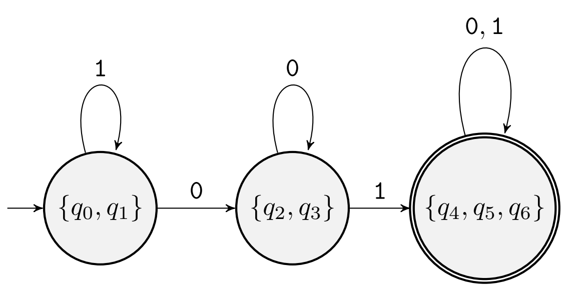
מחלקות השקילות של $\equiv_A^0$ הן $\{q_0, q_1, q_2, q_3\}$ ו- $\{q_4, q_5, q_6\}$.
מחלקות השקילות של $\equiv_A^1$ הן $\{q_0, q_1\}$, $\{q_2, q_3\}$ ו- $\{q_4, q_5, q_6\}$. למשל,

$$q_4 \equiv_A^0 q_5 \text{ and } q_5 = \delta(q_4, 0) \equiv_A^0 \delta(q_5, 0) = q_5$$
$$q_4 = \delta(q_4, 1) \equiv_A^0 \delta(q_5, 1) = q_6$$

ולכן $q_4 \equiv_A^1 q_5$, כלומר q_4 ו-q_5 נמצאים באותה מחלקת שקילות ביחס ל-$\equiv_A^1$. בנוסף,

$$q_0 \equiv_A^0 q_2 \text{ and } \delta(q_0, 1) = q_1 \notin F$$
$$\delta(q_2, 1) = q_4 \in F$$

לכן $q_0 \not\equiv_A^1 q_2$. מכאן ש-q_0 ו-q_2 מופיעים במחלקת שקילות נפרדת ביחס ל-$\equiv_A^1$.
עדיין לא הגענו לנקודת שבת, ולכן נמשיך בתהליך ונגלה כי מחלקות השקילות של $\equiv_A^2$ הן $\{q_0, q_1\}$, $\{q_2, q_3\}$ ו- $\{q_4, q_5, q_6\}$.
הגענו לנקודת שבת, כלומר, $i^* = 1$. אוטומט מינימלי השקול ל-A נראה כך:



איור 7.3: אוטומט סופי דטרמיניסטי A' מינימלי

7.2 שפות חסרות-הקשר

בשבועות הקודמים הצגנו את המודל של אוטומטים סופיים, וראינו שהשפות שהאוטומטים האלה מזהים הם בדיוק השפות הרגולריות. אנו פונים כעת להצגת כלי חדש ועשיר יותר לתיאור שפות.
נפתח בדוגמה לדקדוק חסר הקשר (context-free grammar) שיסומן ב-G (מלשון grammar):

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

הדקדוק מאופיין באמצעות:

- משתנים (variables) – A ו-B. בדרך כלל יסומנו באותיות גדולות.

- טרמינלים (terminals) – $0, 1, \#$. בדרך כלל מתוארים באמצעות אותיות קטנות, מספרים וסימנים מיוחדים.
- חוקי גזירה (derivation rules) – נכתבים בצורה $\alpha \to \beta$. בדוגמה דלעיל: $\{A \to 0A1, A \to B, B \to \#\}$.
- משתנה התחלתי – המוסכמה היא שהמשתנה בצד שמאל של הכלל הראשון שנכתב בדקדוק הוא המשתנה ההתחלתי. בדוגמה: A הוא המשתנה ההתחלתי.

הדקדוק G גוזר (derives) מילה. למשל, הדקדוק G גוזר את המילה 00#11 ע״י סדרת הגזירה הבאה:

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$$

הסימון ״$\Rightarrow$״ מתאר גזירה. מילה מורכבת מטרמינלים בלבד ולא יכולה להכיל משתנים.
השפה של הדקדוק G מכילה את כל המילים שנגזרות מהמצב ההתחלתי של G, ומסומנת $L(G)$. במקרה שלנו: $L(G) = \{0^n\#1^n : n \geq 0\}$.
שימו לב שהשפה $L(G)$ איננה שפה רגולרית, ולכן ״כוח ההבעה״ של דקדוקים חסרי-הקשר גדול יותר מזה של אוטומטים סופיים. שפות חסרות הקשר צמחו מהצורך לעבד שפה טבעית, ואנו נטפל בשפות חסרות הקשר מבחינה תיאורטית.

הגדרה 7.6 (דקדוק חסר-הקשר). **דקדוק חסר-הקשר** (CFG) הוא רביעייה $G = \langle V, \Sigma, R, S \rangle$ כאשר:

- V היא קבוצת משתנים.
- Σ היא אלפבית ומתקיים $\Sigma \cap V = \emptyset$. בהקשר של דקדוקים, האותיות ב-Σ נקראים גם טרמינלים.
- R היא קבוצה של חוקי גזירה, כאשר חוק גזירה הוא מחרוזת מהצורה: $V \to (V \cup \Sigma)^*$. משמע: משתנה בודד בצד שמאל של החץ ומחרוזת כלשהי שמורכבת מטרמינלים ומשתנים בצד ימין של החץ.

הערה 7.7. הגדרות רחבות יותר של דקדוקים מתירות גם לצד שמאל של חוק הגזירה להכיל טרמינלים. זה אומר שניתן להשתמש בכלל הגזירה רק אם המשתנה שבצד שמאל שלו מופיע בהקשר המתאים ביחס לטרמינלים. בהגדרה שלנו אנו לא מתירים טרמינלים בצד שמאל ולכן ניתן להשתמש בכלל גזירה עבור משתנה מסוים בלי להתחשב בהקשר שבו הוא נמצא (איזה טרמינלים מופיעים לידו). מכאן התקבל השם ״דקדוק חסר-הקשר״.

- S הוא משתנה התחלתי שמקיים $S \in V$.

דוגמה 7.8. נתבונן בדקדוק חסר-הקשר G הבא:

$$\begin{aligned} A &\to 0A0 \\ A &\to 1A1 \\ A &\to \varepsilon \end{aligned}$$

לעתים קרובות מקצרים את הכתיבה ומשתמשים בסימון | כדי לתאר מספר חוקי גזירה בשורה אחת:

$$A \to 0A0|1A1|\varepsilon$$

מהי השפה של G? נפעיל מספר פעמים כללי גזירה מ-G כדי לקבל ״תחושה״ למילים שנגזרות מ-G:

$$A \Rightarrow 0A0 \Rightarrow 00A00 \Rightarrow 001A100 \Rightarrow 001100$$

נסיק כי: $L(G) = \left\{w \in \{0,1\}^* : w \text{ is a palindrome of even length}\right\}$. כמובן שנדרשת כאן הוכחה פורמלית (צריך להוכיח שתי הכלות), אבל נדלג על הפורמליזם.
אם היינו חפצים בפלינדרומים מעל $\{0,1\}$ באורך כלשהו (לאו דווקא זוגי) היינו משתמשים בדקדוק:

$$A \to 0A0|1A1|\varepsilon|0|1$$

הגדרה 7.9. יהי $G = \langle V, \Sigma, R, S \rangle$ דקדוק חסר-הקשר. אם $u, v, w \in (V \cup \Sigma)^*$ ו- $A \to w$ הוא חוק בדקדוק, נאמר כי uAv **מייצר את (yields)** uwv, ונרשום $uAv \Rightarrow uwv$.
נאמר ש-u **גוזרת את (derives)** v, ונרשום $u \overset{*}{\Rightarrow} v$ אם קיימת סדרה סופית $u_1, u_2, \ldots, u_k \in (V \cup \Sigma^*)$, $k \geq 1$, ומתקיים:

$$u = u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k = v$$

השפה של דקדוק G מסומנת $L(G)$ ומוגדרת ע״י: $L(G) := \left\{w \in \Sigma^* : S \overset{*}{\Rightarrow} w\right\}$. שימו לב שהמילים ב-$L(G)$ מכילים רק טרמינלים.

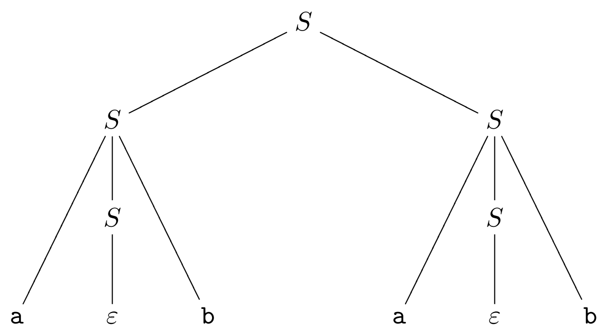
דוגמה 7.10. נתבונן בדקדוק $G=\langle\{S\},\{\mathtt{a},\mathtt{b}\},R,S\rangle$ כאשר:

$$R:S\to\mathtt{a}S\mathtt{b}|SS|\varepsilon$$

האם המילה abab בשפה של G? התשובה חיובית, הנה סדרה שגוזרת אותה:

$$S\Rightarrow SS\Rightarrow\mathtt{a}S\mathtt{b}S\Rightarrow\mathtt{ab}S\Rightarrow\mathtt{aba}S\mathtt{b}\Rightarrow\mathtt{abab}$$

לעתים נוח יותר לתאר סדרת גזירה באמצעות **עץ גזירה (derivation tree)** שנראה כך:



כאשר כל קודקוד שאינו עלה מייצג משתנה בדקדוק ועלים מייצגים טרמינלים.
אם נחליף בדקדוק את a בסוגר שמאלי "(" ואת b בסוגר ימני ")", נקבל את שפת הסוגריים המקוננות החוקיות.
השפה $L(G)$ אינה רגולרית. הדרך הפשוטה לראות זאת היא להבחין שאם $L(G)$ הייתה רגולרית אז גם השפה $L(G)\cap\mathtt{a}^*\mathtt{b}^*=\{\mathtt{a}^n\mathtt{b}^n:n\geq 0\}$ הייתה רגולרית לפי תכונות הסגור של שפות רגולריות, בסתירה לטענה 5.1.

דוגמה 7.11. כיצד נגדיר דקדוק חסר-הקשר ששפתו היא השפה $(\mathtt{a}\cup\mathtt{b})^*\,\mathtt{a}\,(\mathtt{a}\cup\mathtt{b})^*$? אפשרות אחת היא הדקדוק:

$$S\to\mathtt{a}S|\mathtt{b}S|S\mathtt{a}|S\mathtt{b}|\mathtt{a}$$

אפשרות נוספת היא "לחקות" את הביטוי הרגולרי:

$$S\to A\mathtt{a}A$$
$$A\to\mathtt{a}A|\mathtt{b}A|\varepsilon$$

כאשר המשתנה A מתאים לחלק $(\mathtt{a}\cup\mathtt{b})^*$ בביטוי הרגולרי דלעיל.

הגדרה 7.12 (שפה חסרת-הקשר)**.** שפה $\mathscr{L}$ תיקרא **שפה חסרת-הקשר** (CFL) אם קיים דקדוק חסר-הקשר G שעבורו $L(G)=\mathscr{L}$.

הערה 7.13. לא כל השפות הן חסרות-הקשר. למשל, השפה $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n:n\geq 0\}$ אינה שפה חסרת-הקשר. (ההוכחה בדוגמה 9.1).

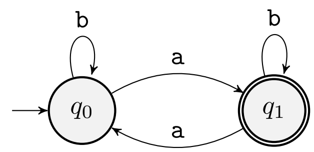
משפט 7.14. כל שפה רגולרית היא שפה חסרת-הקשר.

הוכחה. בהינתן DFA $A=\langle Q,\Sigma,q_0,\delta,F\rangle$ נמיר אותו לדקדוק חסר-הקשר $G=\langle V,\Sigma,R,S\rangle$ כך שמתקיים $L(G)=L(A)$. נגדיר את הדקדוק G באופן הבא:

- $V:=\{V_q:q\in Q\}$, כל משתנה בדקדוק מתאים למצב באוטומט.
- נוסיף את החוק $V_q\to\sigma V_{q'}$ לדקדוק אם"ם $\delta(q,\sigma)=q'$ עבור $q,q'\in Q$ ו-$\sigma\in\Sigma$.
- לכל מצב מקבל $q\in F$ נוסיף את החוק $V_q\to\varepsilon$.
- $S:=V_{q_0}$.

על מנת להוכיח כי אכן $L(G)=L(A)$ עלינו להראות שכל ריצה של האוטומט משרה גזירה של המילה, ולהיפך. (המשך ההוכחה הושאר כתרגיל.) $\square$

דוגמה 7.15. נתבונן באוטומט הסופי הבא:



תוצאת ההמרה שלו לדקדוק חסרת הקשר היא:

$$V_{q_0} \to \mathtt{a}V_{q_1}|\mathtt{b}V_{q_0}$$
$$V_{q_1} \to \mathtt{a}V_{q_0}|\mathtt{b}V_{q_1}|\varepsilon$$

8 שבוע 4 - 03.04.19

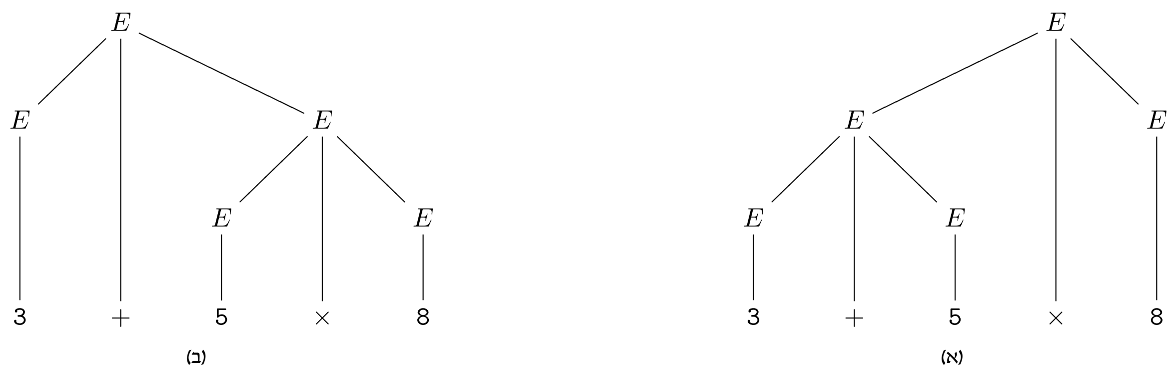
8.1 ריבוי משמעות בשפות חסרות הקשר

הגדרה 8.1. נאמר על דקדוק חסר-הקשר G שהוא **רב-משמעי (ambiguous)** אם קיימת מילה עם שני עצי גזירה שונים.

דוגמה 8.2. נתבונן בדקדוק חסר-הקשר שמוגדר באופן הבא:

$$G = \langle \{E\}, \{+, \times, 0, 1, \ldots, 9\}, R, E \rangle$$
$$E \to E + E | E \times E | 0 | 1 | \ldots | 9$$

המשתנה E נקרא כך מלשון expression. נטען ש-G הוא דקדוק רב-משמעי ע״י כך שנציג שני עצי גזירה שונים עבור המילה $3 + 5 \times 8$:



איור 8.1: שני עצי גזירה שונים עבור המילה $3 + 5 \times 8$

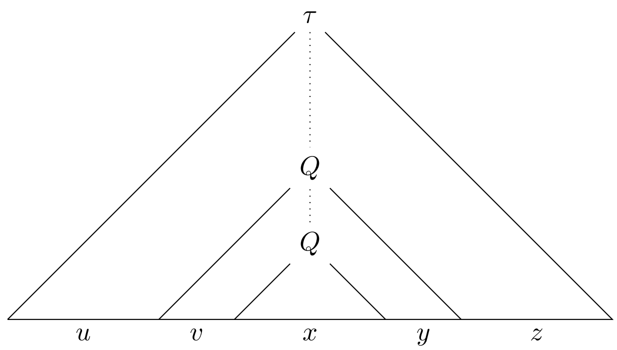
8.2 למת הניפוח לשפות חסרות הקשר

לשפות חסרות הקשר יש גם תכונת ״ניפוח״ כמו לשפות רגולריות. בהקשר של שפות רגולריות השתמשנו בלמת הניפוח כדי להוכיח ששפה אינה רגולרית – כך גם נעשה במקרה של שפות חסרות הקשר.

משפט 8.3 (למת הניפוח לשפות חסרות הקשר)**.** אם L שפה חסרת-הקשר אז קיים $p \geq 1$ (קבוע הניפוח) כך שלכל מילה $w \in L$ שעבורה $|w| \geq p$ קיימת חלוקה $w = uvxyz$ שמקיימת את התנאים הבאים:

- $|vy| > 0$. לא ייתכן שגם $y = \varepsilon$ וגם $v = \varepsilon$.
- $|vxy| \leq p$.
- לכל $i \geq 0$, מתקיים $uv^ixy^iz \in L$.

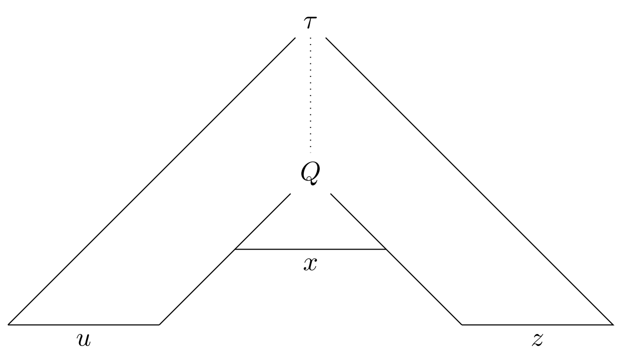
הוכחה. רעיון ההוכחה הוא שאם המילה מספיק ארוכה אז גובה העץ הגזירה עבורה יהיה מספיק גבוה כך שתתרחש חזרה של משתנה במסלול משורש לעלה. הבה נפרמל רעיון זה.
הואיל ו-L שפה חסרת-הקשר, קיים דקדוק חסר הקשר $G = \langle V, \Sigma, R, S \rangle$ ששפתו היא L. יהי b האורך המקסימלי של צד ימין של חוק ב-G (זכרו שצד ימין של חוק בדקדוק הוא מילה ב-$(V \cup \Sigma)^*$ ולכן ניתן לדבר על אורך). אזי בעץ גזירה של מילה כלשהי מ-G לכל קודקוד יש לכל היותר b בנים.
נגדיר $p := b^{|V|+1}$, ותהי $w \in L$ כך ש- $|w| \geq p$. גובה עץ גזירה עבור w מכיל לפחות $b^{|V|+1}$ עלים ולכן גובהו הוא לפחות $|V| + 1$. (זכרו שבעץ b-ארי בגובה h יש לכל היותר b^h עלים.)
נסמן ב-τ עץ גזירה עבור w עם מספר קודקודים מינימלי. כיוון שהגובה של τ הוא לפחות $|V| + 1$, קיים מסלול מהשורש לעלה שיש בו לפחות $|V| + 2$ קודקודים. נקבע מסלול כזה. היות שעלה בעץ גזירה הוא טרמינל, הרי שמסלול משורש לעלה עם לפחות $|V| + 2$ קודקודים מכיל לפחות $|V| + 1$ משתנים. לפי עיקרון שובך היונים, יש לפחות משתנה אחד, Q, שמופיע פעמיים במקטע האחרון במסלול שאורכו לפחות $|V| + 1$.
נגדיר חלוקה של w ל- $uvxyz$ כמו באיור:



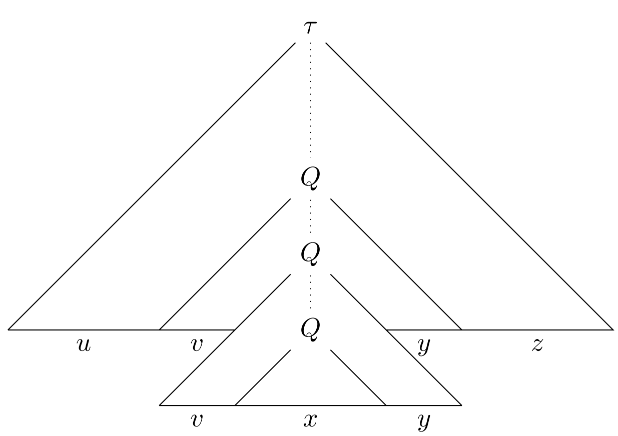
כאשר x היא המחרוזת שנגזרת מ-Q התחתון ואילך, vxy היא המחרוזת שנגזרת מ-Q העליון ו-$uvxyz$ היא המחרוזת שנגזרת משורש העץ.

נוודא שעבור החלוקה שהוגדרה מתקיימים תנאי הלמה:

- $|vy| > 0$, כי אם $v = \varepsilon = y$ אז יש ל-w עץ גזירה עם מספר קודקודים קטן מ-τ, בסתירה לבחירת τ.
- $|vxy| \leq p$, כי בחרנו את המשתנה Q במקטע האחרון במסלול שאורכו לפחות $|V| + 1$. המופע העליון של Q הוא בגובה לכל היותר $|V| + 1$, ולכן
$$|vxy| \leq b^{|V|+1} = p$$
- נטען שלכל $i \geq 0$ יש עץ גזירה ל- uv^ixy^iz, ומכך ינבע $uv^ixy^iz \in L$.
עבור $i = 0$:



עבור $i \geq 2$ נחזור על התהליך שמתואר באיור:



וביתר פורמליות, אנו למעשה טוענים כי: $Q \overset{*}{\Rightarrow} x$ וגם $Q \overset{*}{\Rightarrow} vQy$.

$\square$

9 שבוע 5 - 08.04.19

9.1 שימוש בלמת הניפוח לשפות חסרות הקשר

כיצד נוכל להיעזר בלמה 8.3 כדי להוכיח ששפה מסוימת אינה חסרת הקשר? באופן דומה לשלילת למת הניפוח עבור שפות רגולריות, נאמץ את הניסוח הבא: אם לכל $p \geq 1$ קיימת מילה $w \in L$ עם $|w| \geq p$ כך שלכל חלוקה של w ל-$uvxyz$ המקיימת $|vy| > 0$ ו-$|vxy| \leq p$ קיים $i \geq 0$ כך שהמילה $uv^ixy^iz \notin L$ אז L אינה שפה חסרת הקשר.

דוגמה 9.1. נטען כי השפה $L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$ אינה שפה חסרת-הקשר.
בהינתן $p \geq 1$ נתבונן במילה $w = a^pb^pc^p \in L$. בוודאי מתקיים $|w| \geq p$. תהי $w = uvxyz$ חלוקה כלשהי של w המקיימת $|vy| > 0$ וגם $|vxy| \leq p$. מכיוון ש- $|vxy| \leq p$ אחת מהאפשרויות הבאות חייבות להתקיים:

- $vxy \in \mathtt{a}^+$
- $vxy \in \mathtt{a}^+\mathtt{b}^+$
- $vxy \in \mathtt{b}^+$
- $vxy \in \mathtt{b}^+\mathtt{c}^+$
- $vxy \in \mathtt{c}^+$

שימו לב שלא ייתכן כי $vxy \in \mathtt{a}^+\mathtt{b}^p\mathtt{c}^+$ שהרי $|vxy| \leq p$. עבור $i = 0$, המילה $uv^ixy^iz = uxz$ לא נמצאת בשפה, משום שבכל אחד מן המקרים אנו מקפחים לפחות אות אחת. לפיכך נסיק: L אינה שפה חסרת-הקשר.

9.2 שפות חסרות הקשר אינן סגורות לחיתוך

טענה 9.2. CFL לא סגורה לחיתוך.

הוכחה. נראה שתי שפות חסרות הקשר L_1, L_2 כך שהחיתוך שלהן $L_1 \cap L_2 = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$, שאינו חסר הקשר. נגדיר:

$$L_1 \coloneqq \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m : n, m \geq 0\}$$
$$L_2 \coloneqq \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n : n, m \geq 0\}$$

וניווכח שאכן:

$$L_1 \cap L_2 = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$$

נותר להראות ש- $L_1 \in \mathsf{CFL}$ וגם $L_2 \in \mathsf{CFL}$. נציג דקדוק חסר-הקשר עבור L_1 (הדקדוק עבור L_2 דומה):

$$S \to TC$$
$$T \to \mathtt{a}T\mathtt{b}|\varepsilon$$
$$C \to \mathtt{c}C|\varepsilon$$

מכאן נובע: $L_1 \in \mathsf{CFL}$ וגם $L_2 \in \mathsf{CFL}$ אבל $L_1 \cap L_2 \notin \mathsf{CFL}$ ולכן CFL לא סגורה לחיתוך. □

9.3 צורה נורמלית של חומסקי

הגדרה 9.3. נאמר על דקדוק חסר-הקשר שהוא ב**צורה הנורמלית של חומסקי** אם כל כלליו הם מהצורה:

$$A \to BC$$
$$A \to a$$

כאשר a הוא טרמינל ו-A, B ו-C הם משתנים וגם $B, C \neq S$. בנוסף, אנו מאפשרים את הכלל $S \to \varepsilon$, כאשר S הוא המשתנה ההתחלתי.

משפט 9.4. לכל שפה חסרת-הקשר קיים דקדוק חסר-הקשר בצורה נורמלית של חומסקי שמזהה אותה.

הוכחה. נציג תהליך שיעביר כל דקדוק חסר-הקשר לדקדוק חסר-הקשר שקול (במובן שהם מזהים את אותה שפה) שהוא בצורה נורמלית של חומסקי. התהליך מורכב מחמישה שלבים, כאשר בעת ביצוע כל צעד אנו משמרים את השפה המזוהה ע"י הדקדוק.
נסמן ב-S את המשתנה ההתחלתי בדקדוק. לצורך תיאור התהליך: משתנים יתוארו באמצעות אותיות לטיניות גדולות $A, B, \ldots$, טרמינלים באמצעות אותיות לטיניות קטנות $a, b, \ldots$ ושילוב של משתנים וטרמינלים באותיות יווניות קטנות $\alpha, \beta, \ldots$.

1. שלב ראשון: מחליפים את המשתנה ההתחלתי במשתנה חדש S_0 ומוסיפים את החוק $S_0 \to S$. זה מבטיח שצד ימין של כל חוק בדקדוק לא מכיל את המשתנה ההתחלתי.

2. שלב שני: נפטרים מגזירות ε שאינן מ-S.

$$\begin{matrix} A \to \alpha B\beta \\ B \to \varepsilon \end{matrix} \Longrightarrow A \to \alpha\beta|\alpha B\beta$$

שימו לב שאם $A \to \alpha B\beta B\gamma$ (יש כמה מופעים של B בצד ימין) אז נוסיף את החוקים $A \to \alpha\beta B\gamma$, $A \to \alpha B\beta\gamma$ וגם $A \to \alpha\beta\gamma$, שהרי אנחנו מחליפים כל מופע של B בצד ימין של חוק ב-ε. אם יש חוק מהצורה $A \to B$ אז נוסיף את החוק $A \to \varepsilon$ ונפעיל עליו את התהליך רקורסיבית (בתנאי שלא הסרנו כבר כלל מהצורה הזאת).

3. שלב שלישי: נפטרים מכללים שבהם צד ימין הוא משתנה/טרמינל יחיד.

$$\begin{matrix} A \to \alpha B\beta \\ B \to \gamma \end{matrix} \Longrightarrow A \to \alpha\gamma\beta$$

4. שלב רביעי: משנים כללים בהם בצד ימין יש יותר משני משתנים/טרמינלים.

$$A \to \alpha_1\alpha_2\ldots\alpha_k \Longrightarrow \begin{matrix} A \to \alpha_1 A_1 \\ A_1 \to \alpha_2 A_2 \\ \vdots \\ A_{k-2} \to \alpha_{k-1}\alpha_k \end{matrix}$$

כאשר כל α_i הוא משתנה או טרמינל יחיד ו- $k \geq 3$. המשתנים $\{A_i\}_{i=1}^{k-2}$ הם משתנים חדשים שאנו יוצרים.

5. שלב חמישי: מעבירים את כל גזירות הטרמינלים לצורה $V_a \to a$.

$$A \to aB \Longrightarrow \begin{matrix} A \to V_a B \\ V_a \to a \end{matrix}$$

□

דוגמה 9.5. נתבונן בדקדוק הבא שמייצר את כל המילים באורך זוגי מעל $\{0, 1\}$:

$$S \to 00S|01S|10S|11S|\varepsilon$$

נעביר אותו לצורה נורמלית של חומסקי:

$$S \to 00S|01S|10S|11S|\varepsilon \Longrightarrow \begin{matrix} S \to 0A|0B|1A|1B|\varepsilon \\ A \to 0S \\ B \to 1S \end{matrix} \Longrightarrow \begin{matrix} S \to V_0A|V_0B|V_1A|V_1B|\varepsilon \\ A \to V_0S \\ B \to V_1S \\ V_0 \to 0 \\ V_1 \to 1 \end{matrix}$$

9.4 חיתוך שפות רגולריות עם שפות חסרות הקשר

לעתים קרובות כאשר מציגים אלגוריתם מסוים על דקדוקים חסרי-הקשר או שמעוניינים להוכיח תכונה כלשהי לגביהם, נוח להניח שהם ניתנים בצורה נורמלית של חומסקי. דוגמה כזו נראה בהוכחת המשפט החשוב הבא:

משפט 9.6. אם $L_1 \in \mathsf{CFL}$ ו- $L_2 \in \mathsf{REG}$ אז $L_1 \cap L_2 \in \mathsf{CFL}$.

הוכחה. בהינתן דקדוק חסר-הקשר $G = \langle V, \Sigma, R, S \rangle$ בצורה נורמלית של חומסקי עבור L_1 ואוטומט סופי דטרמיניסטי $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F \rangle$ עבור L_2 נבנה דקדוק G' עבור השפה $L_1 \cap L_2$, כלומר $L(G') = L(G) \cap L(\mathcal{A})$.
נגדיר $G' = \langle V', \Sigma, R', S' \rangle$ באופן הבא:

- $V' = (Q \times V \times Q) \cup \{S'\}$. כל משתנה מלבד S' הוא מהצורה $[pAq]$ כאשר $p, q \in Q$ ו- $A \in V$. הרעיון הוא שמתוך משתנה $[pAq]$ הדקדוק G' יגזור מילים שגם נגזרות מהמשתנה A וגם מובילות מהמצב p למצב q באוטומט $\mathcal{A}$.
- נתאר את קבוצת החוקים R'.

  טיפול במצב ההתחלתי: לכל $q_f \in F$ נוסיף את החוק $S' \to [q_0 S q_f]$. אם $S \to \varepsilon$ וגם $q_0 \in F$ אז נוסיף גם את החוק $S' \to \varepsilon$.

  לכל חוק $A \to BC$ בדקדוק G ולכל מצב ביניים $s \in Q$ נוסיף את החוק:

$$[pAq] \to [pBs][sCq]$$

  לכל חוק $A \to a$ נוסיף את החוק $[pAq] \to a$ אם ורק אם $\delta(p, a) = q$.

  (שימו לב שטיפלנו בכל החוקים בדקדוק G כי הוא בצורה נורמלית של חומסקי.)

נטען כי לכל מילה $w \in \Sigma^*$ עם $|w| \geq 1$ מתקיים $[pAq] \overset{*}{\Rightarrow}_{G'} w$ אם ורק אם $A \overset{*}{\Rightarrow}_G w$ וגם $\delta(p, w) = q$.
את הטענה נוכיח באינדוקציה על אורך המילה.
בסיס: $|w| = 1$. נסמן $w = a$. לפי הבניה מתקיים:

$$[pAq] \overset{*}{\Rightarrow}_{G'} a \iff \begin{array}{c} A \to a \\ \delta(p, a) = q \end{array}$$

ולכן הטענה נכונה עבור מקרה הבסיס. עבור צעד האינדוקציה, נניח נכונות הטענה עבור $|w| < n$ ונניח את נכונותה עבור $|w| = n$.
לפי הבניה $[pAq] \overset{*}{\Rightarrow}_{G'} w$ אם ורק אם השתמשנו בחוק מהצורה $[pAq] \to [pBs][sCq]$ בגזירה של w מ-G'. אזי קיימת חלוקה של w ל-xy כאשר $[pBs] \overset{*}{\Rightarrow}_{G'} x$ וגם $[sCq] \overset{*}{\Rightarrow}_{G'} y$. המילים x ו-y קצרות יותר מ-w, ולכן לפי הנחת האינדוקציה מתקיים:

$$[pBs] \overset{*}{\Rightarrow}_{G'} x \iff \begin{array}{c} B \overset{*}{\Rightarrow}_G x \\ \delta(p, x) = s \end{array}$$

וגם:

$$[sCq] \overset{*}{\Rightarrow}_{G'} y \iff \begin{array}{c} C \overset{*}{\Rightarrow}_G y \\ \delta(s, y) = q \end{array}$$

בהסתמך על החוק $A \to BC$ מתקיים $A \overset{*}{\Rightarrow}_G xy = w$. בנוסף, $\delta(p, xy) = q$, כנדרש.
כדי לסיים את ההוכחה, אנו טוענים כי לכל $w \in \Sigma^*$, $w \in L(G') \iff w \in L(G) \wedge w \in L(\mathcal{A})$.
אם $w = \varepsilon$, אכן אנו מקבלים $S' \to \varepsilon$ אם ורק אם $S \to \varepsilon$ וגם $q_0 \in F$ אם ורק אם $\varepsilon \in L(G)$ וגם $\varepsilon \in L(\mathcal{A})$. אחרת, $|w| \geq 1$ והטענה נובעת מהטענה הקודמת. $\square$

10 שבוע 5 - 10.04.19

כעת אנו מתחילים את החלק השני של הקורס: חישוביות. המודל המתמטי של מחשב שעליו נתבסס נקרא "מכונת טיורינג" (Turing machine) והוא פותח ע"י Alan Turing בשנת 1936.

10.1 מכונת טיורינג

המודל של מכונת טיורינג מזכיר את המודל של אוטומטים סופיים – גם הוא מכיל מצבים ומעברים בין מצבים. יחד עם זאת, קיימים הבדלים מהותיים בין המודלים:

- המודל של מכונת טיורינג תומך בכמות זיכרון לא מוגבלת. מילת הקלט עבור המכונה כתובה על גבי סרט (tape) אינסופי. אם המילה סופית, אז בסיומה מצויים תאים ריקים (להלן: רווחים, blanks) שיסומנו בסמל _.
- הראש הקורא של המכונה רשאי לזוז ימינה ושמאלה על גבי הסרט ללא הגבלה.
- המכונה רשאית לכתוב (ולא רק לקרוא) על גבי הסרט.
- מכונת טיורינג מכילה שני מצבים מיוחדים: מצב קבלה ומצב דחיה. אם בשלב כלשהו בריצת המכונה הגענו למצב מקבל או מצב דוחה, המכונה עוצרת. אם לא הגענו בשום שלב למצב מקבל או מצב דוחה, המכונה ממשיכה לרוץ ללא הגבלה.

לפני שניתן את ההגדרה הפורמלית של המודל, נפתח את האינטואיציה בעזרת דוגמה.

דוגמה 10.1. נרצה לתאר מכונת טיורינג עבור השפה $L = \left\{w\#w : w \in \{0,1\}^*\right\}$. נתבונן במילת הקלט הבאה:

0	0	0	1	#	0	0	0	1	_	_	...

נתאר את פעולת המכונה באופן מילולי באמצעות אלגוריתם:

1. סרוק את הסרט עד שהגעת ל-_. אם אין # או יש לפחות שתי #, דחה את המילה.
2. זגזג בין מיקומים תואמים ובדוק שיש בהם את אותה אות. אם התו הנוכחי איננו #:
   (א) מחק את האות הנוכחית (=כתוב במקומה את האות x) וזכור מהי.
   (ב) התקדם ימינה עד לתא הלא מחוק הראשון מימין לתו #.
   (ג) אם בתא יש רווח _, המילה נדחית. אם בתא יש אות שונה מזו שזכרנו, המילה נדחית. אחרת, יש בתא את האות שזכרנו, ומוחקים אותה (הופכים ל-x).
   (ד) חזור שמאלה ל-x הימני ביותר (הראשון שנתקל בו) משמאל לתו #.
   (ה) התקדם צעד ימינה.
   (ו) חזור על שלב 2.
3. התקדם ימינה עד לתא הראשון מימין ל-# שאין בו x. אם יש בו _, קבל את המילה. אחרת, דחה את המילה.

כך ייראו השלבים הראשונים (מצב הראש הקורא/כותב מודגש):

0	0	0	1	#	0	0	0	1	_	_	...
x	0	0	1	#	0	0	0	1	_	_	...
x	0	0	1	#	**0**	0	0	1	_	_	...
x	0	0	1	#	**x**	0	0	1	_	_	...
x	**0**	0	1	#	x	0	0	1	_	_	...

$\vdots$

הגדרה 10.2 (מכונת טיורינג). **מכונת טיורינג (מ"ט)** היא שביעייה $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}\rangle$, כאשר:

- Q – קבוצת מצבים (סופית).
- Σ – א"ב הקלט. לא מכיל את התו רווח _. בדוגמה: $\{0, 1, \#\}$.

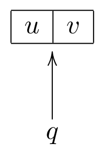
- Γ – א״ב העבודה. $\_ \in \Gamma$ וגם $\Sigma \subseteq \Gamma$. בדוגמה: $\Gamma = \Sigma \cup \{\_, \mathrm{x}\}$.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ היא פונקציית המעברים.

אם $\delta\left(q, a\right) = \langle q', b, \mathrm{L}\rangle$ אז כאשר המכונה נמצאת במצב q והראש הקורא מצביע על תא שמכיל את האות a, המכונה עוברת למצב q', כותבת b (במקום ה-a) ומזיזה את הראש הקורא תא אחד שמאלה (L מציין תזוזה שמאלה ו-R מציין תזוזה ימינה).

- $q_0 \in Q$ – מצב התחלתי.
- $q_{acc} \in Q$ – מצב מקבל.
- $q_{rej} \in Q$ – מצב דוחה.

מכונת טיורינג לא-דטרמיניסטית מוגדרת באותו אופן, חוץ משינוי בפונקציית המעברים: $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}}$, ובמקום מצב התחלתי יחיד אנו נטפל בקבוצה של מצבים התחלתיים $Q_0 \subseteq Q$.

קונפיגורציות **קונפיגורציה** של מכונת טיורינג מאופיינת ע״י מצב נוכחי, תוכן הסרט ומיקום הראש הקורא. מייצגים קונפיגורציה באמצעות המחרוזת uqv כאשר $q \in Q$ ו- $u, v \in \Gamma^*$. המחרוזת uqv מייצגת את הקונפיגורציה שבה המצב הנוכחי הוא q, תוכן הסרט הוא uv (מקובל לחשוב על v ללא רווחים), והראש הקורא מצביע על האות הראשונה של v.



הקונפיגורציה ההתחלתית עבור מילת קלט $w \in \Sigma^*$ היא $q_0 w$.
נאמר על הקונפיגורציה uqv שהיא קונפיגורציה **עוצרת** (halting) אם $q = q_{acc}$ או $q = q_{rej}$. אם $q = q_{acc}$ נאמר גם שהקונפיגורציה **מקבלת**, ואם $q = q_{rej}$ נאמר כי הקונפיגורציה **דוחה**.
יהיו $a, b, c \in \Gamma$, $u, v \in \Gamma^*$ ו- $q, q' \in Q$. **הקונפיגורציה העוקבת** של $uaqbv$ היא:

- אם $\delta\left(q, b\right) = \langle q', c, \mathrm{L}\rangle$ אז הקונפיגורציה העוקבת היא $uq'acv$.
- אם $\delta\left(q, b\right) = \langle q', c, \mathrm{R}\rangle$ אז הקונפיגורציה העוקבת היא $uacq'v$.

אם הקונפיגורציה היא qav ומתקיים $\delta\left(q, a\right) = \langle q', b, \mathrm{L}\rangle$ (כלומר, רוצים ״ליפול״ מתחילת הסרט), אז הקונפיגורציה העוקבת מוגדרת להיות $q'bv$ (דריכה במקום).

11 שבוע 6 - 15.04.19

11.1 מכונת טיורינג - המשך

הגדרה 11.1 (ריצה מקבלת של מכונת טיורינג). בהינתן מילה $w \in \Sigma^*$, **ריצה מקבלת של מכונת טיורינג** M על w היא סדרת קונפיגורציות סופית $C_0, C_1, \ldots, C_m$ שמקיימת:

1. C_0 היא קונפיגורציה התחלתית ($q_0 w$).
2. לכל $0 \le i < m$ הקונפיגורציה C_{i+1} עוקבת לקונפיגורציה C_i.
3. C_m היא קונפיגורציה מקבלת.

הגדרה 11.2 (שפה של מכונת טיורינג). **השפה של מכונת טיורינג** M, שמסומנת $L(M)$, היא קבוצת המילים $w \in \Sigma^*$ שקיימת עליהן ריצה מקבלת.

הגדרה 11.3 (המחלקה RE). נאמר ששפה $\mathscr{L}$ היא Recursively Enumerable (או: ניתנת לזיהוי ע״י מכונת טיורינג) אם קיימת מכונת טיורינג M כך ש- $L(M) = \mathscr{L}$. במצב כזה נאמר כי M **מזהה** את $\mathscr{L}$. סימון: $\mathscr{L} \in \mathsf{RE}$.

הערה 11.4. מכונת טיורינג M לא בהכרח עוצרת על קלטים שאינם ב-$L(M)$. יש שלושה ״גורלות״ לריצה של מכונת טיורינג על מילה מסוימת: מקבלת, דוחה או לא עוצרת. ייתכן גם כי M לא עוצרת ולא חוזרת על קונפיגורציה פעמיים.

הגדרה 11.5 (המחלקה R). נאמר ששפה $\mathscr{L}$ היא Recursive אם קיימת מכונת טיורינג M כך ש- $L(M) = \mathscr{L}$ וגם M עוצרת על כל קלט. במצב כזה נאמר כי M **מכריעה** את $\mathscr{L}$. סימון: $\mathscr{L} \in \mathsf{R}$.

הערה 11.6. אם M מכריעה את $\mathscr{L}$ אז M מזהה את $\mathscr{L}$. (ההפך לא בהכרח נכון)

דוגמה 11.7. נגדיר $L := \left\{0^{2^n} \middle| n \ge 0\right\}$, ונטען כי $L \in \mathsf{R}$, כלומר, נציג מכונת טיורינג M שמכריעה את L.
הרעיון הוא לאפיין רקורסיבית את המילים שבשפה L. נגדיר פרדיקט good על המספרים הטבעיים באופן הבא:

$$\mathsf{good}(k) \overset{\text{def}}{\iff} \exists n \ge 0,\ k = 2^n$$

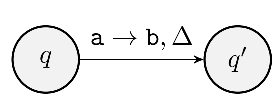
נשים לב שלכל $k \in \mathbb{N}$ מתקיים: $\mathsf{good}(k) \iff k = 1 \vee \mathsf{good}\left(\frac{k}{2}\right)$. נוכח איפיון רקורסיבי זה נבנה מכונת טיורינג שתכריע את L. נגדיר מכונת טיורינג M שבהינתן מילת קלט $w \in \Sigma^*$ תפעל כך:

1. אם הסרט ריק, דחה.
2. החלף את ה-0 הראשון ברווח _, על-מנת שנוכל לזהות את תחילת הסרט.
3. סרוק את הסרט משמאל לימין, מחק כל 0 שני (הפוך אותו ל-x). צעד זה מסיר כמחצית ממספר האפסים.
4. אם בקלט נשאר 0 יחיד, קבל.
5. אם בקלט נשאר יותר מ-0 יחיד וגם נשאר מספר אי-זוגי של 0-ים, דחה.
6. חזור עם הראש הקורא לתחילת הסרט (מסומן ברווח _) ובצע את שלב 3.

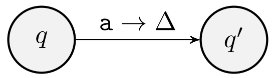
נתאר את המכונה במדויק. $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej} \rangle$, כאשר:

- $Q := \{q_1, q_2, q_3, q_4, q_5, q_{acc}, q_{rej}\}$
- $\Sigma := \{0\}$
- $\Gamma := \{0, \_, \mathtt{x}\}$
- $q_0 := q_1$

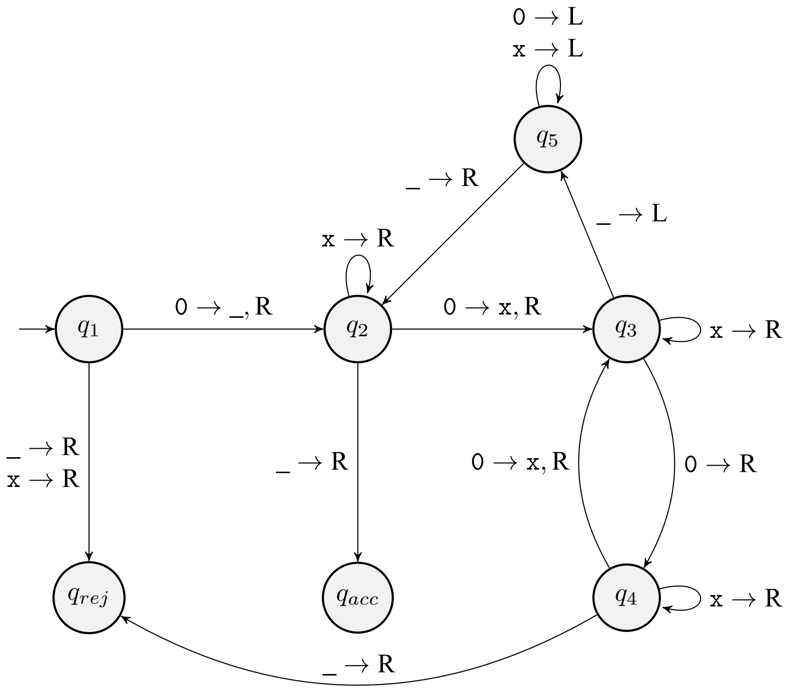
את δ נתאר באמצעות דיאגרמת מצבים. באיור דיאגרמת מצבים למכונת טיורינג נאמץ את המוסכמה הבאה. אם $\delta(q, \mathtt{a}) = \langle q', \mathtt{b}, \Delta \rangle$ כאשר $\Delta \in \{\mathrm{L}, \mathrm{R}\}$ נכתוב:



אם $\mathsf{a} = \mathsf{b}$ מקצרים ורושמים:



הנה דיאגרמת המצבים עבור δ:



כאשר מתארים מעבר ממצב מסוים למצב עוצר (q_{acc} או q_{rej}) אין חשיבות לצד (L, R) שאליו מעבירים את הראש הקורא. כך למשל, במעבר מ-q_1 ל-q_{rej} יכולנו לרשום $\_ \to \mathrm{L}$. שכנעו את עצמיכם שהמכונה המתוארת מבצעת בדיוק את השלבים שהוסברו קודם לכן.

הגדרה 11.8 (המחלקה co-RE). $\overline{\mathscr{L}} \in \mathsf{RE} \overset{\text{def}}{\iff} \mathscr{L} \in \mathsf{co\text{-}RE}$, כלומר, $\mathscr{L} \in \mathsf{co\text{-}RE}$ אם ורק אם קיימת מכונת טיורינג שמזהה את $\overline{\mathscr{L}}$.

משפט 11.9. $\mathsf{R} = \mathsf{RE} \cap \mathsf{co\text{-}RE}$.

הוכחה. ראשית נוכיח $\mathsf{R} \subseteq \mathsf{RE} \cap \mathsf{co\text{-}RE}$. תהי $L \in \mathsf{R}$. לפי הגדרה 11.5, קיימת מכונת טיורינג M שמכריעה את L. לכן, קיימת מכונת טיורינג (אותה M) שמזהה את L. לפי הגדרה 11.3 אנו מקבלים $L \in \mathsf{RE}$. כשם שהוכחנו שהשפות הרגולריות סגורות תחת משלים (ראו משפט 3.1), כך גם R סגורה למשלים – אם נחליף בין q_{acc} ו-q_{rej} במכונה M נקבל מכונת טיורינג $\overline{M}$ שעוצרת על כל קלט ומזהה את $\overline{L}$ (לשון אחר: $\overline{M}$ מכריעה את $\overline{L}$). בפרט, $\overline{M}$ מזהה את $\overline{L}$ ולכן $\overline{L} \in \mathsf{RE}$. לפי הגדרה 11.8 אנו מקבלים $L \in \mathsf{co\text{-}RE}$. לפיכך, $L \in \mathsf{RE} \cap \mathsf{co\text{-}RE}$.
נוכיח את ההכלה בכיוון ההפוך $\mathsf{RE} \cap \mathsf{co\text{-}RE} \subseteq \mathsf{R}$. תהי $L \in \mathsf{RE} \cap \mathsf{co\text{-}RE}$. אזי נתונה מכונת טיורינג M_1 שמזהה את L ונתונה מכונת טיורינג M_2 שמזהה את $\overline{L}$. עלינו לשלב בדרך כלשהי את פעולת שתי המכונות כדי לבנות מכונת טיורינג M שמכריעה את L. הרעיון הוא להריץ במקביל את M_1 ו-M_2. אם M_1 קיבלה את מילת הקלט, אז M גם תקבל. אם M_2 קיבלה את מילת הקלט, אז M תדחה. היות שמילת הקלט נמצאת ב-L או ב-$\overline{L}$, מובטח שנעצור ונחזיר תשובה נכונה. כעת נסביר את המשמעות של "הרצה במקביל". לכל $i = 1, 2, 3, \ldots$ המכונה M תבצע:

1. הרצה של M_1 על מילת הקלט i צעדים; אם M_1 קיבלה, עצור וקבל.
2. הרצה של M_2 על מילת הקלט i צעדים; אם M_2 קיבלה, עצור ודחה.
3. הגדל את i.

$\square$

11.2 Enumerator

ספרן (Enumerator) הוא מודל מתמטי ששקול למכונת טיורינג. ספרן E הוא מכונת טיורינג עם מדפסת וללא קלט התחלתי. E מדפיסה מילים באמצעות המדפסת והיא עשויה להדפיס את אותה מילה מספר פעמים (אפילו ∞). השפה של E מוגדרת כך: $L\left(E\right)=\left\{ w\in\Sigma^{*}:E\text{ prints }w\right\}$. המודל הזה שימושי כיוון שהוא מאפיין את המחלקה RE, כפי שמראה המשפט הבא:

משפט 11.10. $L\in\mathsf{RE}\iff$ קיים ספרן E שעבורו $L\left(E\right)=L$.

הוכחה. $(\Rightarrow)$: בהינתן ספרן E עבור L נבנה מכונת טיורינג שמזהה את L. בהינתן מילת קלט $w\in\Sigma^{*}$, M תפעל כך:

> M מריצה את E. בכל פעם ש-E מדפיסה מילה $y\in\Sigma^{*}$, M בודקת האם $y=w$. אם כן, M עוצרת ומקבלת. אחרת, E ממשיך לרוץ והבדיקה ממשיכה.

$(\Leftarrow)$: נתונה מכונת טיורינג שמזהה את L. נרצה לבנות ספרן E שעבורו $L\left(E\right)=L$. נזכור ש-Σ^{*} היא קבוצת בת-מניה ולכן נוכל לסדר את כל המילים $w_{1},w_{2},w_{3},\ldots$ (למשל בסדר לקסיקוגרפי). הרעיון הנאיבי: הרצת M על כל אחת מן המילים ולהדפיס מילה אם M קיבלה אותה. רעיון זה לא יעבוד שהרי ייתכן ש-M לא תעצור על מילה מסוימת.
הרעיון המוצלח: עבור $j=1,2,3,\ldots$ נריץ את M j צעדים על כל המילים $w_{1},w_{2},\ldots,w_{j}$. אם M מקבלת מילה, נדפיס אותה.
אנו טוענים שהאלגוריתם נכון – כלומר, $L\left(E\right)=L$. אם $w\in L\left(E\right)$ אז w התקבלה ע"י M אחרי מספר סופי של צעדים, ולכן $w\in L\left(M\right)=L$. בכיוון ההפוך, אם $w\in L\left(M\right)=L$ אז M עוצרת ומקבלת את w אחרי i_{1} צעדים. קיים אינדקס i_{2} כך ש-$w_{i_{2}}=w$, ולכן החל מהאיטרציה ה-$\max\left\{ i_{1},i_{2}\right\}$, E תדפיס את w. כלומר, $w\in L\left(E\right)$. $\square$

12 שבוע 7 - 29.04.19

12.1 הבעיה העשירית של הילברט

ב-8 באוגוסט 1900, בוועידת פריז של הקונגרס הבינלאומי של המתמטיקאים, הציג המתמטיקאי הגרמני דויד הילברט רשימה של 23 בעיות פתוחות במתמטיקה. הבעיה העשירית של הילברט:

לתאר אלגוריתם שמכריע האם לפולינום במספר משתנים עם מקדמים שלמים קיימים שורשים שלמים.

למשל, לפולינום $6x^3yz^2 + 3xy^2 - x^3 - 10$ כן קיים שורש שלם: $x = 5, y = 3, z = 0$.
נשאלת השאלה מהו אלגוריתם. הילברט אמר על אלגוריתם: "תהליך שאיתו ניתן להכריע אחרי מספר סופי של פעולות". בהינתן התיאור האינטואיטיבי הזה של המושג אלגוריתם, לא ברור כיצד ניתן **להוכיח** שלא קיים אלגוריתם שפותח בעיה כלשהי.
ב-1936 יצאו לאור מאמרים של אלונזו צ'רץ' ואלן טיורינג שעסקו בין היתר בהגדרת המושג "אלגוריתם". **תזת צ'רץ'-טיורינג** היא השערה במדעי המחשב שקובעת כי "קיים אלגוריתם" $\Longleftrightarrow$ "כריע ע"י מכונת טיורינג". חשוב להבין שתזת צ'רץ'-טיורינג לא מהווה משפט מתמטי (כי לא ברור מה זה אלגוריתם).
עם התקדמות בתורת החישוביות ובמתמטיקה, הוכיח המתמטיקאי יורי מאטיאשביץ' בשנת 1970 שלא קיים אלגוריתם שמכריע את הבעיה העשירית של הילברט.

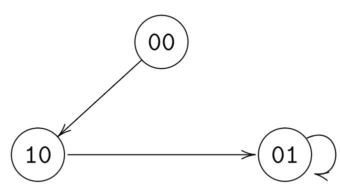
12.2 תיאור אלגוריתמים

בהמשך הקורס נתעניין בכריעות של בעיות, ולשם כך נצטרך לתאר אלגוריתמים בעזרת מכונות טיורינג. אנו מבחינים בין שלוש דרכים לתאר אלגוריתמים:

1. תיאור פורמלי של מכונת טיורינג, הכולל תיאור מלא של פונקציית המצבים.
2. תיאור של מימוש מכונת טיורינג – כיצד הראש הקורא מתנהג ואיזה מידע נכתב לסרט.
3. תיאור אבסטרקטי – תיאור האלגוריתם בשפה טבעית. כך נהגנו בקורס "אלגוריתמים".

דוגמה 11.7 מהווה המחשה לתיאור פורמלי של מכונת טיורינג וגם לתיאור של מימוש.
הקלט למכונת טיורינג הוא מילה. אולם, מנסיוננו, אנו מכירים אלגוריתמים שמקבלים כקלט אובייקטים מורכבים יותר כגון: גרפים, פולינומים, וקטורים וכדומה. על מנת שמכונת טיורינג תוכל לממש אלגוריתם על אובייקט מורכב, עלינו לקודד את האובייקט למילה. נשתמש בסימון $\langle A \rangle$ על מנת לתאר מילה שמהווה את **הקידוד** של אובייקט A.

דוגמה 12.1 (קידוד גרף)**.** נתבונן בגרף המכוון



קידוד אפשרי של הגרף הוא לרשום בתחילת המילה את הקודקודים מופרדים ב-# ולאחר מכן את הצלעות מופרדות ב-$:

00#01#10#$00#10$10#01$01#01$

דוגמה 12.2. נרצה לתאר מכונת טיורינג עבור השפה

$$L \coloneqq \{\langle G \rangle : G \text{ is connected undirected graph}\}$$

נתחיל בתיאור אבסטרקטי של האלגוריתם המקבל כקלט גרף $G = (V, E)$ וקודקוד התחלתי v_0:

$C \coloneqq \emptyset$
$T \coloneqq \{v_0\}$
while $T \neq \emptyset$:

  pop t from T
  for all $v \in V \setminus (C \cup T)$, $v \neq t$:

```
        if there is an edge {v,t} in E then:
            add v to T
    add t to C
accept if and only if C = V, otherwise, reject
```

הקבוצה C מכילה קודקודים שמחוברים ל-v_0 וכבר נבדקו; הקבוצה T מכילה קודקודים שמחוברים ל-v_0 וטרם נבדקו. המשתנה t מייצג את הקודקוד "הפעיל" שנבדק כעת.

נראה כעת שניתן גם לתאר את האלגוריתם ברמת המימוש של מכונת טיורינג.

א"ב הקלט יוגדר ע"י $\Sigma = \{0, 1, \#, \$\}$, כאשר אנו עובדים עם הקידוד הבא עבור גרף עם n קודקודים ו-m צלעות:

$$v_1\#v_2\#\ldots\#v_n\#\$v_{i_1}\#v_{j_1}\$\ldots\$v_{i_m}\#v_{j_m}\$$$

א"ב העבודה יוגדר ע"י $\Gamma = \Sigma \cup \{0,1\} \times \{C,T,A\}$. שימו לב שהזוג $(0,C)$ מהווה אות אחת בא"ב העבודה שאותה נרשום בקיצור 0_C. האותיות C, T ו-A מייצגות קודקוד שנמצא ב-C, ב-T ואת הקודקוד הפעיל t (active) באלגוריתם האבסטרקטי, בהתאמה.

כדי לסמן שייכות של קודקוד מסוים לקבוצה/תפקיד C, T או t אנו נסמן את הביט הראשון שלו ב-subscript המתאים.

תיאור פעולת המכונה:

1. סמן את הביט הראשון של הקודקוד ב-T. כלומר, החלף את 0 ב-0_T ואת 1 ב-1_T.
2. כל עוד יש קודקודים שמסומנים ב-T:
   (א) סמן ב-A את הביט הראשון של קודקוד שמסומן ב-T (החלף 0_T ב-0_A ואת 1_T ב-1_A).
   (ב) סרוק את רשימת הקודקודים. אם יש קודקוד שאינו מסומן ב-C או ב-T או ב-A, בדוק האם יש צלע בינו לבין הקודקוד הפעיל. אם יש, סמן אותו ב-T.
   (ג) נסמן את הקודקוד הפעיל ב-C.
3. אם כל הקודקודים מסומנים ב-C, קבל. אחרת, דחה.

הערה 12.3. שימו לב שבתיאור מכונת טיורינג בדוגמה האחרונה הנחנו בצורה סמויה שהקידוד הוא חוקי. כאשר עובדים עם קידודים, על המכונה לבדוק שהקידוד חוקי – ואם הוא לא חוקי, על המכונה לדחות את המילה. בדרך כלל הקידוד רגולרי ולכן זו לא בדיקה קשה. בנוסף, אנו מבצעים השלמה של שפה ביחס לקידודים נכונים. למשל:

$$L = \{\langle G\rangle : G \text{ is connected undirected graph}\}$$
$$\bar{L} = \{\langle G\rangle : G \text{ is undirected graph, not connected}\}$$

אם לדבר באופן מדויק, המשלים של L הוא כל המילים שהם גרפים לא-מכוונים שאינם קשירים וגם כל המילים שלא מהוות קידודים חוקיים. אולם אנו נאמץ את המוסכמה שאם נתון לנו תחום ייחוס מסוים ואנו מבצעים השלמה, אז ההשלמה מבוצעת במסגרת הקידודים החוקיים.

13 שבוע 7 - 01.05.19

13.1 אי-כריעות

משפט 13.1. קיימת שפה L שעבורה $L \notin \mathsf{R}$.

הוכחה. נראה שתי הוכחות. ההוכחה הראשונה היא משיקולי ספירה. נגדיר שתי קבוצות:

קבוצה A: כל השפות מעל $\Sigma = \{0,1\}$. שימו לב ש-Σ^* בת מניה, ואילו $A = 2^{\Sigma^*}$ בעלת עוצמה א.

קבוצה B: כל מכונות טיורינג מעל א"ב עבודה Σ. לכל מכונת טיורינג יש תיאור סופי בתור מחרוזת, ולכן B בת מניה ($\aleph_0$).

מסקנה: קיימת שפה בקבוצה A כך שאין מכונת טיורינג שמכריעה אותה.
בהוכחה השניה נצביע על שפה שאיננה ב-R. נתבונן בשפה:

$$A_{\mathsf{TM}} = \{(\langle M \rangle, w) : M \text{ accepts } w\}$$

לשם השוואה, השפה $A_{\mathsf{DFA}} = \{(\langle A \rangle, w) : A \text{ accepts } w\}$ כן שייכת ל-R.
תחילה, נשים לב ש-$A_{\mathsf{TM}} \in \mathsf{RE}$. הנה מכונת טיורינג U שמזהה את A_{TM}:

U מריצה את M על w ומחזירה את אותה התשובה ש-M החזירה.

U נקראת "Universal Turing Machine", והוצעה לראשונה ע"י אלן טיורינג ב-1936.

כעת נרצה להראות שלא קיימת מכונת טיורינג שמכריעה את A_{TM}. לשם כך, נניח בשלילה שקיימת מכונת טיורינג H שפועלת כך:

$$H(\langle M \rangle, w) = \begin{cases} \text{accept} & M \text{ accepts } w \\ \text{reject} & M \text{ not accepts } w \end{cases}$$

שימו לב ש-H תמיד עוצרת, אפילו אם M לא עוצרת. בעזרת המכונה H נבנה מכונה D כך:

$$D(\langle M \rangle) = H(\langle M \rangle, \langle M \rangle) = \begin{cases} \text{accept} & M \text{ accepts } \langle M \rangle \\ \text{reject} & M \text{ not accepts } \langle M \rangle \end{cases}$$

ע"י החלפה בין המצב המקבל למצב הדוחה ב-D, נקבל את המכונה $\bar{D}$ שפועלת כך:

$$\bar{D}(\langle M \rangle) = \begin{cases} \text{reject} & M \text{ accepts } \langle M \rangle \\ \text{accept} & M \text{ not accepts } \langle M \rangle \end{cases}$$

נריץ את $\bar{D}$ על הקידוד של עצמה, ונקבל:

$$\bar{D}(\langle \bar{D} \rangle) = \begin{cases} \text{reject} & \bar{D} \text{ accepts } \langle \bar{D} \rangle \\ \text{accept} & \bar{D} \text{ not accepts } \langle \bar{D} \rangle \end{cases}$$

מכאן נובע, $\bar{D}$ לא מזהה נכון את השפה שלה, סתירה. □

הערה 13.2. ההוכחה שראינו מכונה "הוכחה בליכסון". נקבע סידור על מכונות טיורינג $M_1, M_2, M_3, \ldots$ ונתבונן במטריצה הבאה, כאשר בתא ה-(i,j) רשומה תוצאת הפעלת $H(M_i, \langle M_j \rangle)$ (הערכים בטבלה שרירותיים):

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots$	$\langle M_i \rangle$	$\ldots$
M_1	$\boxed{\text{accept}}$	reject	reject			
M_2	accept	$\boxed{\text{reject}}$	accept			
M_3	reject	accept	$\boxed{\text{accept}}$			
$\vdots$				$\ddots$		
M_i					???	
$\vdots$						$\ddots$

שימו לב שתוצאת המכונה D היא בדיוק האלכסון של המטריצה. המכונה $\bar{D}$ מחליפה כל accept ב-reject על האלכסון ולהפך.
הטענה היא שלא קיים i כך ש-$M_i = \bar{D}$. אילו היה קיים i כזה, אז מצד אחד $D(\langle M_i \rangle)$ הוא הערך בתא ה-(i,i) ומצד שני $\bar{D}(\langle M_i \rangle)$ הוא הערך שרשום בתא ה-(i,i). מדובר בסתירה כי $\bar{D}(\langle M_i \rangle) \neq D(\langle M_i \rangle)$ לפי הגדרת המכונה $\bar{D}$.

מסקנה 13.3. $A_{\mathsf{TM}} \notin \mathsf{co\text{-}RE}$, וגם $\overline{A_{\mathsf{TM}}} \notin \mathsf{RE}$.

הוכחה. נזכור ש-$A_{\mathsf{TM}} \in \mathsf{RE}$. אם $A_{\mathsf{TM}} \in \mathsf{co\text{-}RE}$ אז לפי משפט 11.9 היינו מקבלים $A_{\mathsf{TM}} \in \mathsf{R}$, בסתירה למשפט 13.1. מההגדרה של co-RE (הגדרה 11.8) אנו מקבלים כי $\overline{A_{\mathsf{TM}}} \notin \mathsf{RE}$. □

דוגמה 13.4 (בעיית העצירה)**.** נתבונן בשפה הבאה:

$$\mathsf{HALT}_{\mathsf{TM}} = \{(\langle M \rangle, w) : M \text{ halts on } w\}$$

בוודאי $\mathsf{HALT}_{\mathsf{TM}} \in \mathsf{RE}$ שהרי ניתן להשתמש במכונה האוניברסלית U כדי להריץ את M על מילת הקלט ולקבוע כי $(\langle M \rangle, w) \in \mathsf{HALT}_{\mathsf{TM}}$ אם M עצרה.

נראה כעת שאם $\mathsf{HALT}_{\mathsf{TM}} \in \mathsf{R}$ אז $A_{\mathsf{TM}} \in \mathsf{R}$.

טכניקת הוכחה זו ידועה בשם "רדוקציה"

כלומר, נרצה לבנות מכונת טיורינג M_A שמכריעה את A_{TM} בהינתן מכונת טיורינג M_{HALT} שמכריעה את $\mathsf{HALT}_{\mathsf{TM}}$:

> בהינתן קלט $(\langle M \rangle, w)$ המכונה M_A מריצה את M_{HALT} על הקלט $(\langle M \rangle, w)$. אם M_{HALT} דוחה, M_A תדחה גם היא. אם M_{HALT} מקבלת, נריץ את M על w, ונחזיר את אותה התשובה.

שימו לב שאנו מריצים את M על w רק אם M_{HALT} מקבלת, ולכן אנו בטוחים ש-M תעצור על w.
מאחר ש-$A_{\mathsf{TM}} \notin \mathsf{R}$ אנו מסיקים ש- $\mathsf{HALT}_{\mathsf{TM}} \notin \mathsf{R}$.

14 שבוע 8 - 06.05.19

14.1 רדוקציית מיפוי

בדוגמה 13.4 השתמשנו בטכניקת הוכחה שנקראת "רדוקציה" – ראינו שבהינתן מכונת טיורינג M_{HALT} שמכריעה את $\mathsf{HALT_{TM}}$ ניתן לבנות מכונת טיורינג M_A שמכריעה את A_{TM}. מדובר בטכניקת הוכחה כללית וחשובה במדעי המחשב שנפרמל בהרצאה זו.

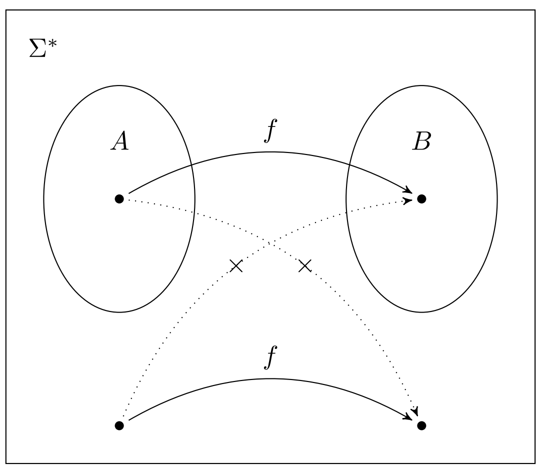
הגדרה 14.1 (פונקציה ניתנת לחישוב)**.** נאמר על פונקציה $f:\Sigma^*\to\Sigma^*$ שהיא **ניתנת לחישוב (computable function)** אם"ם קיימת מכונת טיורינג M_f כך שלכל קלט $w\in\Sigma^*$, המכונה M_f עוצרת עם $f(w)$ על הסרט.

נחשוב על המכונה M_f בתור מכונת טיורינג עם מצב עצירה אחד (במקום שני מצבים עוצרים: מצב מקבל ומצב דוחה).

דוגמה 14.2. נתבונן בפונקציה $f:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ המוגדרת ע"י $f(x,y)=x+y$. אנו טוענים ש-f ניתנת לחישוב. אם x ו-y נתונים בבסיס אונארי אז חישוב $x+y$ הוא בסה"כ שרשור של הייצוגים השונים. למשל, אם $x=2$ ו-$y=3$ אז מילת הקלט ל-M_f תהיה 11#111, ו-M_f תסיר את ה-# ותזיז את החלק הימני צעד אחד שמאלה כדי לקבל 11111 המייצג את המספר $x+y=5$ בבסיס אונארי. אם x ו-y נתונים בבסיס אחר (למשל בסיס בינארי) פעולת החיבור גם ניתנת לחישוב בעזרת אלגוריתם "חיבור ארוך" מבית ספר יסודי.

דוגמה 14.3. נסמן ב-$\mathcal{M}$ את קבוצת כל מכונות הטיורינג בקידוד מסוים. נתבונן בפונקציה $f:\mathcal{M}\to\mathcal{M}$ שבהינתן (קידוד של) מכונת טיורינג $\langle M\rangle$ מחזירה (קידוד של) מכונת טיורינג $\langle M'\rangle$ כך ש-$L(M')=L(M)$, וגם M' לא עוצרת על מילים שאינן ב-$L(M')$. אנו טוענים שהפונקציה f ניתנת לחישוב כאשר המכונה M_f תפעל כך: בהינתן קידוד של M, M_f תוסיף מצב חדש q_{loop} עם חוג עצמי ("בור"). נשנה מעברים שנכנסים ל-q_{rej} כך שייכנסו ל-q_{loop}.

הגדרה 14.4 (רדוקציית מיפוי)**.** שפה $A\subseteq\Sigma^*$ ניתנת **לרדוקציית מיפוי (mapping reduction)** לשפה $B\subseteq\Sigma^*$ אם קיימת פונקציה $f:\Sigma^*\to\Sigma^*$ ניתנת לחישוב כך שלכל מילה $w\in\Sigma^*$ מתקיים: $w\in A\leftrightarrow f(w)\in B$. במקרה כזה נסמן: $A\le_m B$.



איור 14.1: המחשה של רדוקציית מיפוי

הערה 14.5. רדוקציית מיפוי מספקת המרה של שאלות על שייכות ל-A לשאלות על שייכות ל-B. בנוסף, הפונקציה f לא בהכרח פונקציה "על". ניקח דוגמה מעולם המספרים השלמים: אם $A=\{x\in\mathbb{Z}\mid |x|\le 5\}$ ו- $B=\{x\in\mathbb{Z}\mid |x|\le 10\}$ אז הפונקציה $f(x)=2x$ מקיימת את הנדרש (לכל $x\in\mathbb{Z}$ מתקיים $|x|\le 5$ אם ורק אם $|f(x)|\le 10$), אבל היא ממפה רק למספרים השלמים הזוגיים ב-B.

החשיבות של מושג הרדוקציה טמון במשפט הבא:

משפט 14.6 (משפט הרדוקציה)**.** יהיו $A,B\subseteq\Sigma^*$ שפות. אם $A\le_m B$ וגם $B\in\mathsf{R}$ אז $A\in\mathsf{R}$.

הוכחה. הואיל ו- $A\le_m B$ אנו יודעים שלפי הגדרה קיימת פונקציה $f:\Sigma^*\to\Sigma^*$ ניתנת לחישוב כך שלכל $w\in\Sigma^*$ מתקיים $w\in A\leftrightarrow f(w)\in B$. לפי הגדרה של פונקציה ניתנת לחישוב, קיימת מכונת טיורינג M_f שעבור כל קלט $w\in\Sigma^*$ עוצרת עם המילה $f(w)$ על הסרט. הואיל ו-$B\in\mathsf{R}$, קיימת מכונת טיורינג M_B שמכריעה את B.
נבנה מכונת טיורינג M_A שמכריעה את A באופן הבא:

על קלט $w \in \Sigma^*$ תחשב את $f(w)$ באמצעות M_f. לאחר מכן, תריץ את M_B את $f(w)$ ותענה כמוה.

ראשית, M_A עוצרת על כל קלט. שנית נטען כי $L(M_A) = A$. אכן,

$$w \in A \overset{\text{הגדרת}}{\underset{f}{\iff}} f(w) \in B \iff f(w) \in L(M_B) \overset{\text{מאופן פעולת}}{\underset{M_A}{\iff}} w \in L(M_A)$$

מכאן ש-M_A מכריעה את $A \Leftarrow A \in \mathsf{R}$. □

מסקנה ישירה וחשובה ממשפט הרדוקציה שתסייע לנו להוכיח אי-כריעות של שפות:

מסקנה 14.7. אם $A \le_m B$ ו-$A \notin \mathsf{R}$ אז $B \notin \mathsf{R}$.

דוגמה 14.8 (בעיית העצירה באמצעות רדוקציית מיפוי)**.** בבעיית העצירה שראינו בדוגמה 13.4 הוכחנו כי $\mathsf{HALT_{TM}} \notin \mathsf{R}$. כעת נרצה להוכיח את הטענה תוך הפעלת משפט הרדוקציה. פורמלית, אנו נראה כי $A_{\mathsf{TM}} \le_m \mathsf{HALT_{TM}}$. לשם כך אנו נראה שקיימת פונקציה ניתנת לחישוב f מקבוצת הקידודים של $(\langle M \rangle, w)$ כאשר M מ״ט ו-$w \in \Sigma^*$ אל עצמה, אשר מקיימת:

$$(\langle M \rangle, w) \in A_{\mathsf{TM}} \leftrightarrow f(\langle M \rangle, w) \in \mathsf{HALT_{TM}} \qquad (14.1)$$

התבוננו בפונקציה $f(\langle M \rangle, w) = (\langle M' \rangle, w)$ הבאה: f לא משנה את w, אלא מחליפה את המכונה M במכונה M' כך ש-$L(M') = L(M)$ וגם M' לא עוצרת על קלטים שאינם ב-$L(M')$. (כבר ראינו בדוגמה 14.3 שהפונקציה f ניתנת לחישוב.)
אם $(\langle M \rangle, w) \in A_{\mathsf{TM}}$ אז M מקבלת את w, ולפי הבנייה, M' מקבלת את w. בפרט זה אומר ש-M' עוצרת על w ולכן $f(\langle M \rangle, w) = (\langle M' \rangle, w) \in \mathsf{HALT_{TM}}$.
אם $(\langle M \rangle, w) \notin A_{\mathsf{TM}}$ אז M לא מקבלת את w, ולכן M' לא עוצרת על w, כלומר, $(\langle M' \rangle, w) \notin \mathsf{HALT_{TM}}$.
בזאת הוכחנו שמשוואה 14.1 מתקיימת. ממשפט הרדוקציה נובע: $\mathsf{HALT_{TM}} \notin \mathsf{R}$.

דוגמה 14.9 (בעיית העצירה על ε)**.** נתבונן בשפה $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}} = \{\langle M \rangle : M \text{ halts on } \varepsilon\}$. נראה רדוקציה $\mathsf{HALT_{TM}} \le_m \mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$ ומכאן ינבע $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}} \notin \mathsf{R}$. לשם כך, נראה פונקציה ניתנת לחישוב $f(\langle M \rangle, w) = \langle M' \rangle$ כאשר M עוצרת על w אס״ם M' עוצרת על ε. נבנה את M_f כך:

בהינתן $(\langle M \rangle, w)$ נייצר M' באופן הבא:

1. M' כותבת w על הסרט, ומאתחלת את הראש הקורא (מזיזה אותו לתחילת הסרט).
2. M' מריצה את M.

מאחר שבהינתן מילת קלט ε המכונה M' מסמלצת ריצה של M על w, אנו מקבלים כי:

$$\langle M' \rangle \in \mathsf{HALT}^{\varepsilon}_{\mathsf{TM}} \leftrightarrow (\langle M \rangle, w) \in \mathsf{HALT_{TM}}$$

שימו לב שאפילו אם $(\langle M \rangle, w) \in \mathsf{HALT_{TM}}$ לא בהכרח מתקיים ש-M' עוצרת על מילים שאינן ε, שהרי M' דורסת את הקלט שלה עם המילה w – אילו היינו מקבלים מילת קלט ארוכה יותר מ-w הייתה נשארת סיפא של מילת הקלט על גבי הסרט שעלולה הייתה לשבש את החלטת M.
במקרה של $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$ מתקיימת גם הרדוקציה בכיוון ההפוך: $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}} \le_m \mathsf{HALT_{TM}}$. כדי להראות שרדוקציה זו מתקיימת, עלינו להציג פונקציה ניתנת לחישוב $f(\langle M' \rangle) = (\langle M \rangle, w)$ כך ש- M' עוצרת על ε אס״ם M עוצרת על w. לכן, נבחר פשוט $f(\langle M' \rangle) = (\langle M' \rangle, \varepsilon)$ ($M := M'$ ו-$w := \varepsilon$) ו-f מקיימת את הנדרש (וודאו).

משמעות הדוגמה הבאה היא שלא קיים תהליך אוטומטי שמכריע אם שפה מסוימת היא רגולרית:

דוגמה 14.10 (מכונות טיורינג ששפתן רגולרית)**.** נתבונן בשפה

$$\mathsf{REG_{TM}} = \{\langle M \rangle : L(M) \text{ is regular}\}$$

נוכיח כי $\mathsf{REG_{TM}} \notin \mathsf{R}$ ע״י רדוקציה $A_{\mathsf{TM}} \le_m \mathsf{REG_{TM}}$. לשם כך, נרצה להראות שקיימת פונקציה f ניתנת לחישוב

$$f(\langle M \rangle, w) = \langle M' \rangle$$

כך שמתקיים $(\langle M \rangle, w) \in A_{\mathsf{TM}} \leftrightarrow L(M')$ רגולרית.
M' פועלת על מילת קלט $x \in \{0,1\}^*$ כך (M' מוגדרת מעל הא״ב $\{0,1\}$ ללא קשר לא״ב שמעליו מוגדרת M):

1. אם $x \in \{0^n 1^n \mid n \ge 0\}$ אז M' מקבלת את x.
2. אחרת, M' מריצה את M על w ומשיבה כמוה – כלומר, M' מקבלת את x אס״ם M קיבלה את w.

אם $(\langle M \rangle, w) \in A_{\mathsf{TM}}$ אז $L(M') = \{0,1\}^*$ שהרי כל המילים מהצורה $0^n 1^n$ יתקבלו בשלב הראשון, ושאר המילים יתקבלו בשלב השני. מצד שני, אם $(\langle M \rangle, w) \notin A_{\mathsf{TM}}$ אז $L(M') = \{0^n 1^n \mid n \ge 0\}$ שאינה רגולרית.
לפיכך, $(\langle M \rangle, w) \in A_{\mathsf{TM}} \leftrightarrow M' \in \mathsf{REG_{TM}}$, ולפי משפט הרדוקציה נקבל $\mathsf{REG_{TM}} \notin \mathsf{R}$.

15 שבוע 9 - 13.05.19

15.1 רדוקציית מיפוי - המשך

ההוכחה של המשפט הבא דומה להוכחת משפט 14.6:

משפט 15.1 (משפט הרדוקציה ל-RE ו-co-RE). יהיו $A, B \subseteq \Sigma^*$.

1. אם $A \leq_m B$ וגם $B \in \mathsf{RE}$ אז $A \in \mathsf{RE}$.
2. אם $A \leq_m B$ וגם $B \in \text{co-}\mathsf{RE}$ אז $A \in \text{co-}\mathsf{RE}$.

נשים לב שאם $A \leq_m B$ אז $\overline{A} \leq_m \overline{B}$, שהרי אותה פונקציה ניתנת לחישוב f שמעידה כי $A \leq_m B$, מעידה גם ש-$\overline{A} \leq_m \overline{B}$. נסכם זאת בעובדה הבאה:

עובדה 15.2. יהיו $A, B \subseteq \Sigma^*$. אם $A \leq_m B$ אז $\overline{A} \leq_m \overline{B}$.

המשך הדיון בשפה $\mathsf{REG}_{\mathsf{TM}}$ אנו יודעים ש- $A_{\mathsf{TM}} \in \mathsf{RE}$ וגם $A_{\mathsf{TM}} \notin \mathsf{R}$ ולכן $A_{\mathsf{TM}} \notin \text{co-}\mathsf{RE}$. לפי הרדוקציה $A_{\mathsf{TM}} \leq_m \mathsf{REG}_{\mathsf{TM}}$ שהראינו בדוגמה 14.10, אנו מסיקים לפי משפט 15.1 כי $\mathsf{REG}_{\mathsf{TM}} \notin \text{co-}\mathsf{RE}$. כעת נראה שמתקיים $\mathsf{REG}_{\mathsf{TM}} \notin \mathsf{RE}$ ע"י שנראה את הרדוקציה $\overline{A_{\mathsf{TM}}} \leq_m \mathsf{REG}_{\mathsf{TM}}$, או באופן שקול, את הרדוקציה $A_{\mathsf{TM}} \leq_m \overline{\mathsf{REG}_{\mathsf{TM}}}$. מכיוון ש-$A_{\mathsf{TM}} \notin \text{co-}\mathsf{RE}$ אנו נקבל $\overline{\mathsf{REG}_{\mathsf{TM}}} \notin \text{co-}\mathsf{RE}$ ומכאן נסיק $\mathsf{REG}_{\mathsf{TM}} \notin \mathsf{RE}$.
על מנת להראות את הרדוקציה $A_{\mathsf{TM}} \leq_m \overline{\mathsf{REG}_{\mathsf{TM}}}$ עלינו להציג פונקציה ניתנת לחישוב f שעבור קלט ל-A_{TM} מהצורה $(\langle M \rangle, w)$ מחזירה קלט ל-$\mathsf{REG}_{\mathsf{TM}}$ מהצורה $\langle M' \rangle$ כך שמתקיים:

- אם $(\langle M \rangle, w) \in A_{\mathsf{TM}}$ אז $\langle M' \rangle \in \overline{\mathsf{REG}_{\mathsf{TM}}}$, כלומר, $L(M')$ לא רגולרית.
- אם $(\langle M \rangle, w) \notin A_{\mathsf{TM}}$ אז $\langle M' \rangle \notin \overline{\mathsf{REG}_{\mathsf{TM}}}$, כלומר, $L(M')$ רגולרית.

בהינתן קלט $x \in \{0,1\}^*$ המכונה M' תפעל כך:

1. נבדוק האם $x \in \{0^n 1^n : n \geq 1\}$? אם כן, M' מריצה את M על w ומשיבה כמוה.
2. אם לא, M' דוחה את x.

M' עלולה להיתקע בשלב זה

נוודא שאכן מתקיימים התנאים שציפינו מ-M' לקיים:

- אם $(\langle M \rangle, w) \in A_{\mathsf{TM}}$ אז הקלטים היחידים ש-M' מקבלת הם $x \in \{0^n 1^n : n \geq 1\}$, כלומר:
$$L(M') = \{0^n 1^n : n \geq 1\}$$
ולכן $\langle M' \rangle \in \overline{\mathsf{REG}_{\mathsf{TM}}}$.
- אם $(\langle M \rangle, w) \notin A_{\mathsf{TM}}$ אז כל קלט $x \in \{0,1\}^*$ נדחה ע"י M', ולכן $L(M') = \emptyset$. מכאן ש-$L(M')$ רגולרית.

לפיכך הראינו רדוקציית מיפוי $A_{\mathsf{TM}} \leq_m \overline{\mathsf{REG}_{\mathsf{TM}}}$, ונסיק $\mathsf{REG}_{\mathsf{TM}} \notin \mathsf{RE}$.

דוגמה 15.3 (מכונות טיורינג ששפתן אינסופית). נתבונן בשפה $\mathsf{INF}_{\mathsf{TM}} = \{\langle M \rangle : |L(M)| = \aleph_0\}$. נרצה להראות את הרדוקציה
$$A_{\mathsf{TM}} \leq_m \mathsf{INF}_{\mathsf{TM}}$$
ונסיק כי $\mathsf{INF}_{\mathsf{TM}} \notin \text{co-}\mathsf{RE}$. לשם כך עלינו להציג פונקציה ניתנת לחישוב f, שפועלת באופן הבא: $f(\langle M \rangle, w) = \langle M' \rangle$, כאשר $L(M')$ אינסופית $\iff$ M מקבלת את w. המכונה M' תפעל כך:

בהינתן קלט $x \in \{0,1\}^*$: M' מריצה את M על w ומשיבה כמוה.

עתה מתקיים:

- $(\langle M \rangle, w) \in A_{\mathsf{TM}} \Leftarrow L(M') = \{0,1\}^*$.
- $(\langle M \rangle, w) \notin A_{\mathsf{TM}} \Leftarrow L(M') = \emptyset$.

כעת נראה את הרדוקציה $\mathsf{HALT}_{\mathsf{TM}} \leq_m \overline{\mathsf{INF}_{\mathsf{TM}}}$ ומכאן נסיק $\overline{\mathsf{INF}_{\mathsf{TM}}} \notin \text{co-}\mathsf{RE}$. בהינתן $(\langle M \rangle, w)$ נייצר מכונת טיורינג M' כך שמתקיים: M עוצרת על w $\iff$ $L(M')$ סופית. המכונה M' תפעל כך:

בהינתן קלט $x \in \{0,1\}^*$:

1. הרץ את M על w במשך $|x|$ צעדים.
2. אם M עצרה על w במהלך $|x|$ הצעדים – דחה את x. אחרת, קבל את x.

מתקיים:

- $\left(\langle M\rangle, w\right) \in \mathsf{HALT_{TM}} \Leftarrow$ קיים מספר טבעי $\ell \in \mathbb{N}$ כך ש-M עוצרת על w תוך ℓ צעדים. לכן, כל מילת קלט $x \in \{0,1\}^*$ שעבורה $|x| \geq \ell$ תידחה ע״י M'. מכאן ש- $L\left(M'\right) = \left\{x \in \{0,1\}^* : |x| < \ell\right\}$, ובפרט $L\left(M'\right)$ סופית.
- $\left(\langle M\rangle, w\right) \notin \mathsf{HALT_{TM}} \Leftarrow$ במקרה כזה כל מילת קלט תתקבל ע״י M' ולכן $L\left(M'\right) = \{0,1\}^*$ שהיא שפה אינסופית.

15.2 בעיית הריצוף

בשיעור הראשון התוודענו לבעיה חישובית שאיננה כריעה - בעיית הריצוף (מומלץ לרענן את הזיכרון לפני ההגדרות הפורמליות כאן: 1.1). נגדיר את הבעיה פורמלית ונבחן אותה ביסודיות:

קלט:

- קבוצה סופית של k אריחים $T = \{t_1, \ldots, t_k\}$.
- יחס שכנות בין האריחים במאוזן $H \subseteq T \times T$ (horizontal condition).
- יחס שכנות בין האריחים במאונך $V \subseteq T \times T$ (vertical condition).
- אריח התחלתי $t_0 \in T$.

פלט: האם קיים ריצוף חוקי $n \times n$ לכל $n \geq 1$.

ריצוף חוקי: עבור $n \geq 1$, ריצוף חוקי $n \times n$ הוא פונקציה $f : [n] \times [n] \to T$ המקיימת:

- $f(1,1) = t_0$ (מתחילים את הריצוף עם האריח ההתחלתי)
- מתקיימים תנאי שכנות במאוזן: לכל $1 \leq i < n$ ולכל $1 \leq j \leq n$ מתקיים:
$$\left(f(i,j), f(i+1,j)\right) \in H$$
- מתקיימים תנאי שכנות במאונך: לכל $1 \leq i \leq n$ ולכל $1 \leq j < n$ מתקיים:
$$\left(f(i,j), f(i,j+1)\right) \in V$$

דוגמה 15.4 (קלט לבעיית הריצוף)**.** האריחים הם $T = \{1,2,3\}$, עם אריח התחלתי $t_0 = 1$. יחס שכנות במאוזן
$$H = \{(2,1), (1,2), (2,3), (3,1)\}$$
כאשר בזוג (a,b), a מייצג את האריח השמאלי ו-b מייצג את האריח הימני.
יחס שכנות במאונך
$$V = \{(1,2), (2,3), (3,1)\}$$
בזוג (a,b), a מייצג את האריח התחתון ו-b את האריח העליון.

על מנת לדבר על כריעות של בעיה חישובית עלינו להגדיר שפה. במקרה של בעיית הריצוף השפה תוגדר כך:
$$\mathsf{TILE} = \left\{\langle T, H, V, t_0\rangle : \substack{n \geq 1 \text{ לכל} \\ n \times n \text{ קיים ריצוף חוקי}}\right\}$$
תחילה נבחין כי $\mathsf{TILE} \in \text{co-RE}$. נתאר מכונת טיורינג M שמזהה את $\overline{\mathsf{TILE}}$:

עבור $n = 1, 2, \ldots$:

M בודקת האם קיים ריצוף חוקי $n \times n$; אם לא קיים, נכנסת למצב מקבל.
אחרת, ממשיכה לבדוק את ה-n הבא.

כדי לראות ש- $\mathsf{TILE} \notin \mathsf{R}$ אנו נראה את הרדוקציה הבאה $\overline{\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}} \leq_m \mathsf{TILE}$. כלומר, בהינתן $\langle M\rangle$ (קלט ל-$\overline{\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}}$) נרצה לבנות קלט לבעיית הריצוף $\langle T, H, V, t_0\rangle$, כך שמתקיים:
$$M \text{ לא עוצרת על } \varepsilon \iff n \geq 1 \text{ לכל } \text{ קיים ריצוף חוקי } n \times n$$

16 שבוע 9 - 15.05.19

16.1 בעיית הריצוף - המשך

כשהגדרנו את השפה TILE דרשנו שלכל $n \geq 1$ יהיה קיים ריצוף חוקי $n \times n$. תנאי הכרחי ומספיק הוא שקיים ריצוף חוקי לרבע המישור החיובי.

למה 16.1. לכל $n \geq 1$ קיים ריצוף חוקי $n \times n$ $\Longleftrightarrow$ קיים ריצוף חוקי לרבע המישור החיובי.

הוכחה. $(\Rightarrow)$: אם קיים ריצוף חוקי לרבע המישור החיובי, כלומר, קיימת פונקציה $f^* : \mathbb{N} \times \mathbb{N} \to T$ שמקיימת את תנאי הריצוף החוקי, אז בפרט קיימת פונקציה $f_n : [n] \times [n] \to T$ שמקיימת את תנאי הריצוף החוקי. f_n מוגדרת כך:

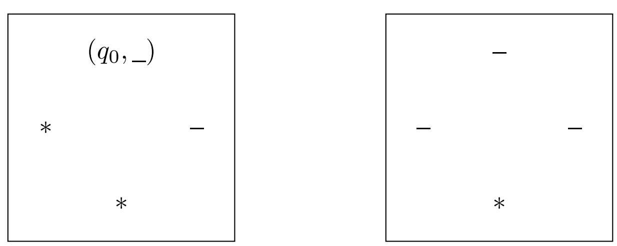
$$f_n\left(i,j\right) \coloneqq f^*\left(i,j\right), \quad \forall i \in [n]\, \forall j \in [n]$$

$(\Leftarrow)$: נניח שלכל $n \geq 1$ קיים ריצוף חוקי $n \times n$ ונרצה להוכיח שקיים ריצוף חוקי לרבע המישור החיובי.
נבנה עץ אינסופי שברמה ה-n-ית שלו יימצאו ריצופים חוקיים מסדר $n \times n$ (אלו יהיו הקודקודים). נחבר צלע בין קודקודים f_n ברמה ה-n ו-f_{n+1} ברמה ה-$n+1$ אם ורק אם f_{n+1} מזדהה עם f_n על קלטים ב-$[n] \times [n]$ (אינטואיטיבית, הריצוף של f_{n+1} הוא המשך חוקי של הריצוף החוקי של f_n).
הואיל ולכל $n \geq 1$ קיים ריצוף חוקי $n \times n$, בכל רמה בעץ יש לפחות קודקוד אחד, ולכן העץ אכן אינסופי. דרגת הפיצול של כל קודקוד היא סופית שהרי יש מספר סופי של אפשרויות להרחיב ריצוף חוקי $n \times n$ לריצוף חוקי $(n+1) \times (n+1)$. שימו לב גם שקודקוד f_{n+1} ברמה ה-$n+1$ חייב להיות מחובר בצלע לקודקוד כלשהו ברמה ה-n (כי ריצוף חוקי $(n+1) \times (n+1)$ מגדיר גם ריצוף חוקי $n \times n$).
כדי לסיים את הטיעון אנו נזכרים **בלמה של קניג**: עץ אינסופי, שכל הדרגות בו סופיות, מכיל מסלול אינסופי היוצא מן השורש. העץ שבנינו מקיים את ההנחות בלמה של קניג כפי שהוסבר לעיל; לפיכך, קיים מסלול אינסופי היוצא מן השורש, או במילים אחרות: קיים ריצוף חוקי לרבע המישור החיובי. $\square$

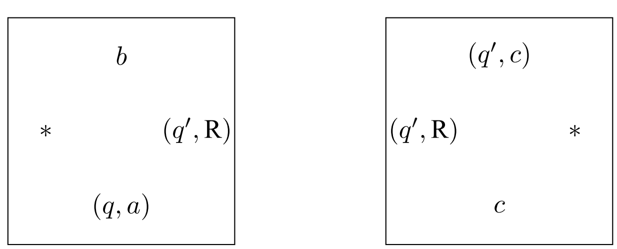
הרעיון מאחורי הרדוקציה $\overline{\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}} \leq_m \text{TILE}$ הוא למדל קונפיגורציות של מכונת טיורינג באמצעות אריחים. אם המכונה M לא עוצרת על ε אז יש אינסוף קונפיגורציות ונוכל לתאר אותם בתור סריג של אריחים שמרצפים את רבע המישור החיובי. אם המכונה M כן עוצרת על ε, אז יש מספר סופי של קונפיגורציות, ונדאג לבחור סוגי מרצפות שלא יאפשרו הרחבה של הריצוף אם הגענו ל-q_{acc} או q_{rej}. בהינתן קלט $\langle M \rangle$ לשפה $\overline{\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}}$ נרצה להחזיר קלט לשפה TILE. הבניה מתוארת לחלוטין באמצעות סוגי המרצפות (תנאי שכנות במאונך ובמאוזן נגזרים מהסימנים שנכתוב על כל אחד מארבעת צדדי המרצפת).

סוגי מרצפות

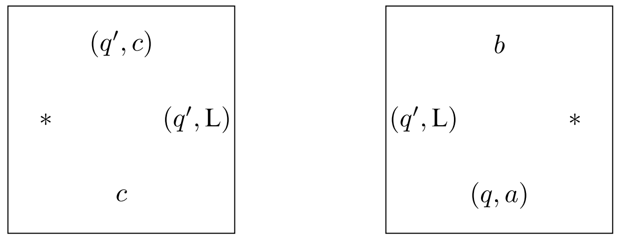
1. מרצפות השורה הראשונה



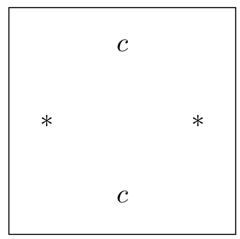
2. לכל מעבר $\delta\left(q,a\right) = \left(q',b,\mathrm{R}\right)$ עבור $q \neq q_{acc}, q_{rej}$ נוסיף את $|\Gamma|+1$ המרצפות הבאות לכל $c \in \Gamma$:



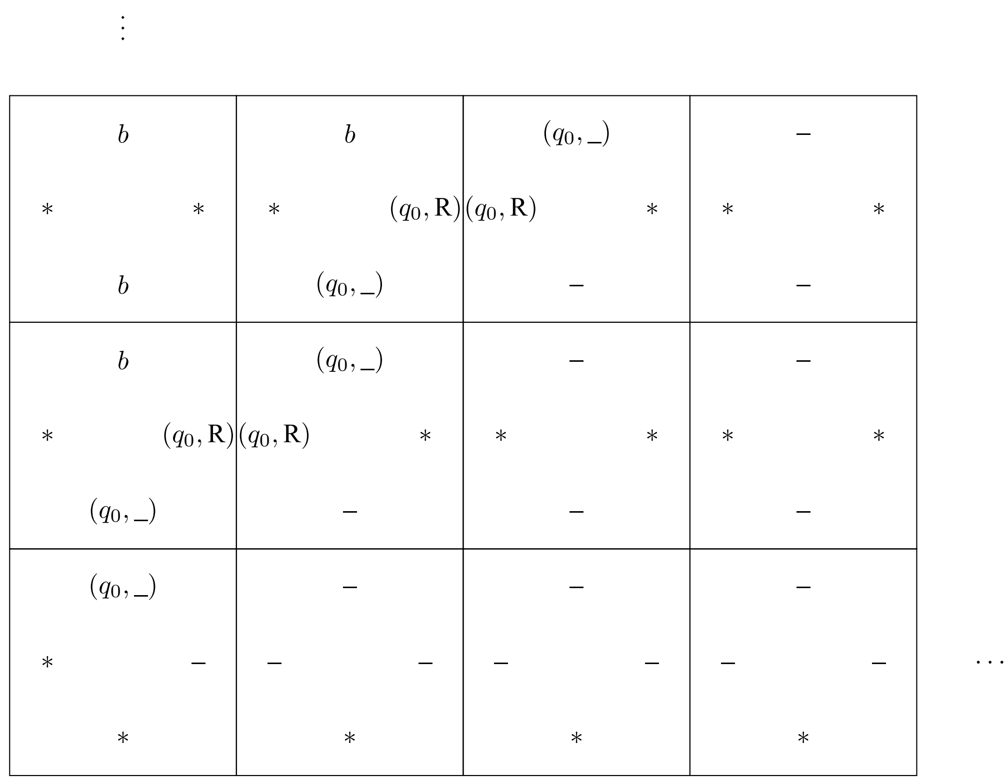
3. באופן דומה, לכל מעבר $\delta\left(q,a\right) = \left(q',b,\mathrm{L}\right)$ עבור $q \neq q_{acc}, q_{rej}$ נוסיף את $|\Gamma|+1$ המרצפות הבאות לכל $c \in \Gamma$:



4. מרצפות ריפוד: לכל $c \in \Gamma$ נוסיף את המרצפות הבאות:



ניתן דוגמה כדי להמחיש את הבניה. עבור מכונת טיורינג עם מצב אחד q_0 ומעבר $\delta\left(q_0, \_\right) = \left(q_0, b, \mathrm{R}\right)$ (רושם b על הסרט וזז ימינה) אנו נייצר את הריצוף הבא:



איור 16.1: ריצוף חוקי שנבנה באמצעות קונפיגורציות של מכונת טיורינג. שימו לב שבחלק העליון של המרצפות בשורה הראשונה מופיעה הקונפיגורציה ההתחלתית. באופן דומה, בחלק העליון של המרצפות בשורה השניה מופיעה הקונפיגורציה השניה וכן הלאה.

נסכם בטבלה את סוגי הסימונים בהגדרת המרצפות.

סימון	מאונך/מאוזן	תפקיד
$*$	מאונך	תחתית השורה הראשונה
$*$	מאוזן	מעברים בין תאים שאינם ביניהם תזוזה של הראש הקורא
_	מאונך	כמו אות רגילה
_	מאוזן	השורה הראשונה (למעט $*$ באריח ההתחלתי)
$c \in \Gamma$	מאונך	סימון התא בסרט
$(q, c) \in Q \times \Gamma$	מאונך	סימון התא בסרט, תיאור מצב ומצביע על הראש הקורא
$(q, \mathrm{D}) \in Q \times \{\mathrm{L}, \mathrm{R}\}$	מאוזן	בין מרצפות שיש ביניהם מעבר של הראש הקורא

טבלה 1: סוגי הסימונים בהגדרת המרצפות

17 שבוע 10 - 20.05.19

17.1 בעיית ההתאמה של פוסט

ניתן דוגמה לבעיה נוספת שאינה כריעה. בעיה זו נקראת בעיית ההתאמה של פוסט (Post correspondence problem), ונסמנה PCP. ניתן להראות את הרדוקציה $A_{\mathsf{TM}} \leq_m$ PCP באופן שמזכיר את הרדוקציה שביצענו לשפה TILE. הנה תיאור של הבעיה:

קלט: קבוצה $\{r_1, r_2, \ldots, r_n\}$ של אבני דומינו מהצורה $r_i = \boxed{\begin{array}{c} w_i^{(u)} \\ \hline w_i^{(d)} \end{array}}$ כאשר $w_i^{(u)}, w_i^{(d)} \in \Sigma^*$ לכל $i \in [n]$.

פלט: האם קיימת התאמה בין המחרוזת שכתובה על החלק העליון של אבני הדומינו לבין המחרוזת שכתובה על החלק התחתון של אבני הדומינו. כלומר, האם קיים $k \in \mathbb{N}$ ואינדקסים $i_1, i_2, \ldots, i_k \in [n]$ כך ש-

$$w_{i_1}^{(u)} \cdot w_{i_2}^{(u)} \cdot \cdots \cdot w_{i_k}^{(u)} = w_{i_1}^{(d)} \cdot w_{i_2}^{(d)} \cdot \cdots \cdot w_{i_k}^{(d)}$$

דוגמה: נתבונן באוסף אבני הדומינו

$$P = \left\{ \boxed{\begin{array}{c} \mathtt{b} \\ \hline \mathtt{ca} \end{array}}, \boxed{\begin{array}{c} \mathtt{a} \\ \hline \mathtt{ab} \end{array}}, \boxed{\begin{array}{c} \mathtt{ca} \\ \hline \mathtt{a} \end{array}}, \boxed{\begin{array}{c} \mathtt{abc} \\ \hline \mathtt{c} \end{array}} \right\}$$

במקרה זה קיימת התאמה:

$$\boxed{\begin{array}{c} \mathtt{a} \\ \hline \mathtt{ab} \end{array}} \boxed{\begin{array}{c} \mathtt{b} \\ \hline \mathtt{ca} \end{array}} \boxed{\begin{array}{c} \mathtt{ca} \\ \hline \mathtt{a} \end{array}} \boxed{\begin{array}{c} \mathtt{a} \\ \hline \mathtt{ab} \end{array}} \boxed{\begin{array}{c} \mathtt{abc} \\ \hline \mathtt{c} \end{array}}$$

המחרוזת שכתובה בחלק העליון של אבני הדומינו a b ca a abc והיא שווה למחרוזת שכתובה בחלק התחתון של אבני הדומינו ab ca a ab c.

17.2 תורת הסיבוכיות

כעת אנו נתמקד במחלקת השפות הכריעות R, וננסה לסווגן לפי כמות המשאבים (זמן, זיכרון, אקראיות) שצריך להשקיע כדי להכריע שפה. נתחיל בדיון על סיבוכיות זמן של אלגוריתמים.

הגדרה 17.1 (המחלקה $\mathsf{TIME}(\cdot)$). עבור פונקציה $t : \mathbb{N} \to \mathbb{N}$ נגדיר את מחלקת סיבוכיות הזמן $\mathsf{TIME}(t(n))$ כך:

$$\mathsf{TIME}(t(n)) = \left\{ L \subseteq \Sigma^* \mid \begin{array}{c} \text{ניתנת להכרעה על-ידי מכונת טיורינג } L \\ \text{דטרמיניסטית בעלת סרט יחיד} \\ O(t(n)) \text{ הרצה בזמן} \end{array} \right\}$$

דוגמה 17.2. נתבונן בשפה $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$. שפה זו ניתנת להכרעה ע״י מכונת טיורינג. מכונת טיורינג שמכריעה את L פועלת בדומה לתהליך שתיארנו בדוגמה 10.1: בהינתן מילה $w \in \{\mathtt{a}, \mathtt{b}\}^*$ המכונה מוחקת a ראשון ו-b ראשון כל עוד נשארו אותיות שלא נמחקו. אם האותיות התרוקנו בו-זמנית, היא מקבלת את w.
ננתח את סיבוכיות הזמן של האלגוריתם המוצע בהנחה שגודל הקלט הוא n (כלומר, $|w| = n$). כל איטרציה עולה $O(n)$ ומספר האיטרציות הוא לכל היותר n; לכן זמן הריצה בסה״כ הוא $O(n^2)$, ולכן $L \in \mathsf{TIME}(n^2)$.
האם אפשר להכריע את L בזמן קצר יותר? כן. הרעיון הוא שבכל איטרציה נמחק חצי מהאותיות (נניח את כל האותיות במקומות הזוגיים). כל איטרציה עולה $O(n)$ ומספר האיטרציות הוא $O(\log_2(n))$ ולכן $L \in \mathsf{TIME}(n \log n)$.
האם אפשר להכריע את L בזמן קצר יותר? לא. המשפט הבא מראה זאת (מובא ללא הוכחה):

משפט 17.3. אם שפה L ניתנת להכרעה בזמן $o(n \log n)$ אז L רגולרית.

נתעכב מעט על שתי דרישות בהגדרת המחלקה $\mathsf{TIME}(t(n))$:

- דרשנו שמכונת טיורינג שמכריעה את L תהיה בעלת **סרט יחיד**: נשים לב ש-$L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ ניתנת להכרעה בזמן לינארי ע״י מכונת טיורינג עם שני סרטים – נעתיק את רצף ה-b-ים לסרט השני ונשווה מספר a-ים למספר b-ים ״במקביל״ עם שני ראשים קוראים. בתרגול נוכיח את המשפט הבא:

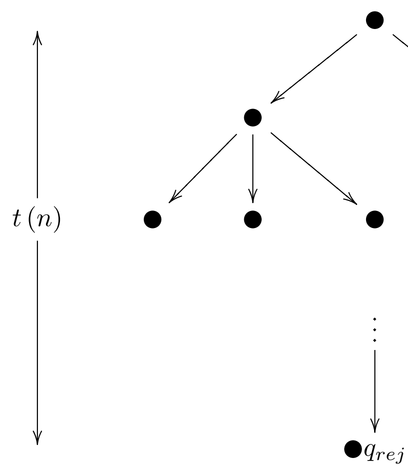
  משפט 17.4. לכל מכונת טיורינג M בעלת k-סרטים שרצה בזמן $t(n)$ קיימת מכונת טיורינג M' שקולה ($L(M) = L(M')$) בעלת סרט יחיד שרצה בזמן $O(t^2(n))$.

- דרשנו שמכונת טיורינג שמכריעה את L תהיה מכונת טיורינג **דטרמיניסטית**. מסתבר שאי-הדטרמיניזם מכניס למשחק מחלקת סיבוכיות זמן חדשה שנסקור כעת.

היזכרו בהגדרה של מכונת טיורינג לא-דטרמיניסטית 10.2.

הגדרה 17.5. נאמר על מכונת טיורינג לא-דטרמיניסטית M שהיא **מכריעה** שפה $\mathscr{L}$ אס"ם $L(M)=\mathscr{L}$ וגם M עוצרת על כל המילים בכל החישובים שלה.

הגדרה 17.6 (זמן ריצה של מכונת טיורינג לא-דטרמיניסטית)**.** תהי N מכונת טיורינג לא-דטרמיניסטית שמכריעה שפה L. זמן הריצה של N הוא פונקציה $t:\mathbb{N}\to\mathbb{N}$, כאשר $t(n)$ הוא מספר הצעדים המקסימלי ש-N מבצעת על קלט באורך n מבין כל המסלולים האפשריים בעץ החישוב של N (גם אם המסלול מסתיים בדחייה).



הגדרה 17.7 (המחלקה $\mathsf{NTIME}(\cdot)$)**.** עבור פונקציה $t:\mathbb{N}\to\mathbb{N}$ נגדיר את מחלקת סיבוכיות הזמן $\mathsf{NTIME}(t(n))$ כך:

$$\mathsf{NTIME}(t(n))=\left\{L\subseteq\Sigma^{*}\mid \begin{array}{c}\text{ניתנת להכרעה על-ידי מכונת טיורינג } L\\ \text{אי-דטרמיניסטית בעלת סרט יחיד}\\ \text{הרצה בזמן } O(t(n))\end{array}\right\}$$

באמצעות המחלקות $\mathsf{TIME}(\cdot)$ ו-$\mathsf{NTIME}(\cdot)$ אנו נגדיר שלוש מחלקות סיבוכיות זמן נוספות:

$$\mathsf{PTIME}:=\bigcup_{k=1}^{\infty}\mathsf{TIME}\left(n^{k}\right)$$
$$\mathsf{NPTIME}:=\bigcup_{k=1}^{\infty}\mathsf{NTIME}\left(n^{k}\right)$$
$$\mathsf{EXPTIME}:=\bigcup_{k=1}^{\infty}\mathsf{TIME}\left(2^{n^{k}}\right)$$

ובמילים: PTIME (בקיצור: P) היא מחלקת כל הבעיות שניתנות להכרעה בזמן פולינומיאלי ע"י מכונת טיורינג דטרמיניסטית;
NPTIME (בקיצור: NP) היא מחלקת כל הבעיות שניתנות להכרעה בזמן פולינומיאלי ע"י מכונת טיורינג אי-דטרמיניסטית;
EXPTIME היא מחלקת כל הבעיות שניתנות להכרעה בזמן אקספוננציאלי ע"י מכונת טיורינג דטרמיניסטית.
משפט אנלוגי למשפט 3.9 מראה שניתן לבצע דטרמיניזציה למכונת טיורינג אי-דטרמיניסטית שכרוך בפיצוץ אקספוננציאלי:

משפט 17.8. לכל מכונת טיורינג אי-דטרמיניסטית הרצה בזמן $O(t(n))$ יש מכונת טיורינג דטרמיניסטית שקולה הרצה בזמן $2^{O(t(n))}$.

ממשפט זה אנו למדים כי:

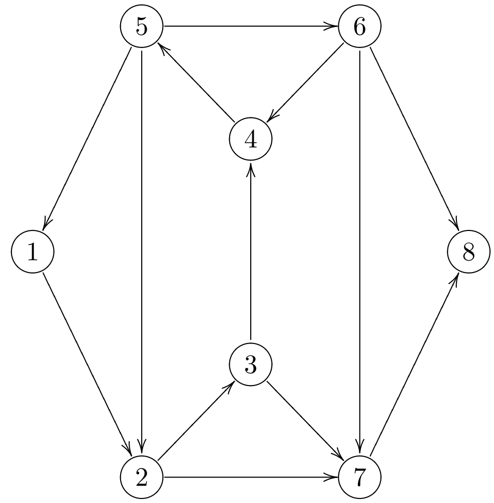
$$\mathsf{PTIME}\subseteq\mathsf{NPTIME}\subseteq\mathsf{EXPTIME}$$

בנוסף, ידוע כי $\mathsf{EXPTIME}\neq\mathsf{PTIME}$. בעיה פתוחה מרכזית במדעי המחשב היא האם $\mathsf{NPTIME}\overset{?}{=}\mathsf{PTIME}$, או בקיצור, האם $\mathsf{NP}\overset{?}{=}\mathsf{P}$. בעיה זו היא אחת מ-7 בעיות המילניום של מכון קליי למתמטיקה, ופותרה יזכה בפרס של מיליון דולר. בהמשך נרחיב מעט על הבעיה.

17.3 בעיית המסלול ההמילטוני

נזכיר כי **מסלול המילטון** בגרף מכוון $G=(V,E)$ הוא מסלול העובר דרך כל קודקודי G פעם אחת בדיוק. בעיית המסלול ההמילטוני היא בהינתן גרף מכוון להכריע האם קיים מסלול המילטון בגרף. למשל בגרף הבא קיים מסלול המילטון (עקבו אחרי

סדר הקודקודים):



אנחנו נסתכל על וריאציה של הבעיה, ונגדיר את השפה הבאה:

$$\text{D-ST-HAMPATH} := \left\{ \langle G, s, t \rangle : \begin{array}{c} G \text{ is a directed graph} \\ s, t \text{ are vertices} \\ \text{there exists Hamiltonian path from } s \text{ to } t \text{ in } G \end{array} \right\}$$

הנה אלגוריתם אקספוננציאלי להכרעת D-ST-HAMPATH:

- נסמן $|V| = n$, $|E| = m$.
- עבור על כל הסדרות של קודקודים באורך n (נחשוב עליהם בתור מילים ב-V^n):
  - בדוק האם מדובר במסלול המילטון שמתחיל ב-s ומסתיים ב-t. ביתר פירוט: בדוק האם המסלול מתחיל בקודקוד s ומסתיים בקודקוד t; בדוק שקודקודים הם עוקבים במסלול אס״ם קיימת קשת מתאימה בגרף; בדוק שכל הקודקודים מופיעים בדיוק פעם אחת.
  - אם מדובר במסלול המילטון חוקי – קבל.
- אם לא קיים מסלול המילטון חוקי – דחה.

אלגוריתם (נאיבי) זה מלמד אותנו כי D-ST-HAMPATH $\in$ EXPTIME.
כעת נראה ש-D-ST-HAMPATH $\in$ NPTIME. מכונת טיורינג לא-דטרמיניסטית N שמכריעה את D-ST-HAMPATH תפעל כך:

- המכונה ״מנחשת״ סדרה של קודקודים באורך n וכותבת אותה על סרט העבודה שלה. (הכוונה ב״ניחוש״ - מבצעת צעדים לא-דטרמיניסטיים שכותבים סדרה של קודקודים על גבי הסרט).
- המכונה בודקת אם סדרת הקודקודים עומדת בתנאים (מתארת מסלול המילטון חוקי שמתחיל ב-s ומסתיים ב-t, כפי שתיארנו באלגוריתם הקודם).
- אם הסדרה עומדת בתנאים – קבל; אחרת – דחה.

נשים לב כי אורך כל ענף חישוב בעץ החישובים של N הוא סופי (כלומר, כל החישובים עוצרים). בנוסף, עומק עץ החישוב הוא פולינומיאלי שהרי ביצענו מספר פולינומיאלי של צעדים כדי לכתוב סדרה של n קודקודים ועוד מספר פולינומיאלי של צעדים לבצע בדיקה האם מדובר במסלול המילטוני או לא. לפיכך, זמן הריצה של N הוא פולינומיאלי, ולכן D-ST-HAMPATH $\in$ NPTIME.
האם D-ST-HAMPATH $\in$ PTIME? לא יודעים.
הבעיה המשלימה $\overline{\text{D-ST-HAMPATH}}$ נמצאת ב-EXPTIME אולם לא יודעים אם היא ב-NPTIME.
מדוע זה חשוב ש-D-ST-HAMPATH $\in$ NPTIME? לומר על בעיה שהיא ב-NPTIME במקום EXPTIME זו אמירה חזקה יותר.
כדי להמחיש מדוע, ניקח דוגמה מעולם האוטומטים. נתבונן בשפות הבאות:

$$A_{\text{DFA}} = \left\{ (\langle A \rangle, w) \mid {}^{A \text{ is DFA}}_{w \in L(A)} \right\}$$
$$A_{\text{NFA}} = \left\{ (\langle A \rangle, w) \mid {}^{A \text{ is NFA}}_{w \in L(A)} \right\}$$

אנו יודעים ש-A_{DFA} ניתנת להכרעה בזמן פולינומיאלי ע״י מכונת טיורינג דטרמיניסטית ולכן $A_{\text{DFA}} \in \text{PTIME}$. בוודאי מתקיים $A_{\text{NFA}} \in \text{EXPTIME}$ כי אפשרות אחת היא לבצע דטרמיניזציה של ה-NFA הכרוכה בפיצוץ אקספוננציאלי ולהפעיל את האלגוריתם שמכריע את A_{DFA} בזמן פולינומיאלי. אפשרות מתוחכמת יותר מבוססת על האבחנה שאם A הוא NFA אז:

$$w \in L(A) \iff \{w\} \cap L(A) \neq \emptyset$$

לכן אם נבנה אוטומט A_w שמזהה את השפה $L(A_w) = \{w\}$ בלבד, נוכל לבצע חיתוך של אוטומטים (בנייה של אוטומט המכפלה) ולהכריע האם השפה של האוטומט שמתקבל היא ריקה או לא (ניתן להכריע זאת ע״י בדיקת ישיגות מהמצב ההתחלתי למצב מקבל). לכן $A_{\text{NFA}} \in \text{PTIME}$.

דוגמה זו ממחישה כי זה לא מופרך לחשוב ש**ייתכן** ש-$\text{P} \stackrel{?}{=} \text{NP}$, אולם מדובר בבעיה פתוחה.

18 שבוע 10 - 22.05.19

18.1 בדיקת ראשוניות

מספר פריק (באנגלית: composite number) הוא מספר שלם חיובי שאפשר לכתוב אותו כמכפלה של שני מספרים שלמים גדולים מ-1. נתבונן בשפה:

$$\mathsf{COMPOSITE} = \{x \in \mathbb{N} \mid x \text{ is composite number}\}$$

אלגוריתם מוכר לפתרון הבעיה הוא בהינתן $x \in \mathbb{N}$ לעבור על כל המספרים $2..\lceil\sqrt{x}\rceil$ ולבצע בדיקה האם הם מחלקים שלמים של x.
אם x נתון כקלט בבסיס אונארי, אז האלגוריתם מבצע $O(x)$ איטרציות וכל איטרציה היא פולינומיאלית בגודל הייצוג של x. לכן, אם x נתון בבסיס אונארי אנו מסיקים $\mathsf{COMPOSITE} \in \mathsf{PTIME}$.
לעומת זאת, אם x נתון כקלט בבסיס בינארי, האלגוריתם שהוצע קודם לכן רץ בזמן אקספוננציאלי בגודל הייצוג של x. כעת נראה ש-$\mathsf{COMPOSITE} \in \mathsf{NPTIME}$. מכונת טיורינג אי-דטרמיניסטית שמכריעה את COMPOSITE מנחשת מספר בין $2..\lceil\sqrt{x}\rceil$ (כותבת אותו בבסיס בינארי על גבי הסרט), ובודקת האם הוא מחלק את x.
אציין כי בשנת 2004 פורסם מאמר ע"י Manindra Agrawal, Neeraj Kayal, Nitin Saxena ממכון IITK בהודו שהראה כי $\mathsf{COMPOSITE} \in \mathsf{PTIME}$. הנה קישור למאמר.

18.2 מוודאים

הגדרה 18.1 (מוודא). **מוודא (verifier)** עבור שפה L הוא מכונת טיורינג דטרמיניסטית V שמקיימת:

$$\left\{w \mid \substack{V \text{ accepts } \langle w,c\rangle \\ \text{for some string } c}\right\} = L$$

המחרוזת c נקראת לפעמים "עד" (witness) והסימון c הגיע מהמילה certificate. **סיבוכיות המוודא** נמדדת ביחס ל-w בלבד.
מוודא פולינומיאלי רץ בזמן פולינומיאלי ב-$|w|$ (בפרט, זה אומר ש-$|c|$ פולינומי ב-$|w|$).

דוגמה 18.2. לשפה D-ST-HAMPATH יש מוודא פולינומיאלי. המוודא מקבל קלט $\langle G,s,t\rangle$ ל-D-ST-HAMPATH ומסלול π בעל n קודקודים. המוודא מקבל את המילה $(\langle G,s,t\rangle, \pi)$ אם ורק אם π הוא מסלול המילטון ב-G שמתחיל ב-s ומסתיים ב-t.
ראינו כבר שבדיקה זו יכולה להתבצע בזמן פולינומיאלי.
גם לשפה COMPOSITE יש מוודא פולינומיאלי. מוודא פולינומיאלי עבור COMPOSITE מקבל מספר טבעי $x \in \mathbb{N}$ ומספר טבעי נוסף $p \in \mathbb{N}$ ובודק האם p מחלק את x. בדיקה זו יכולה להתבצע בזמן פולינומיאלי.
לעומת זאת, לשפה $\overline{\text{D-ST-HAMPATH}}$ לא ידוע האם קיים מוודא פולינומיאלי.

חשיבות המוודאים נעוצה במשפט הבא:

משפט 18.3. תהי L שפה. $L \in \mathsf{NP} \iff$ קיים מוודא פולינומיאלי עבור L.

הוכחה. $(\Rightarrow)$: נניח שקיים מוודא פולינומיאלי V עבור L. כיוון שהמוודא פולינומיאלי, קיים $k \in \mathbb{N}$ כך שזמן הריצה של V הוא לכל היותר $O(n^k)$ (n - גודל הקלט). בהינתן קלט w, מכונת טיורינג לא-דטרמיניסטית עבור L תנחש עד (witness) באורך של עד n^k ותריץ את המוודא על (w,c). אם המוודא קיבל, המכונה תקבל; אחרת, המכונה תדחה.
$(\Leftarrow)$: נניח שקיימת מכונת טיורינג לא-דטרמיניסטית N שרצה בזמן פולינומיאלי ומכריעה את L. נבנה מוודא פולינומיאלי עבור L שיקיים: V מקבל (w,r) אם ורק אם r היא ריצה מקבלת של N על w. הואיל ו-N רצה בזמן פולינומיאלי, גודל הריצה המקבלת $|r|$ הוא פולינומי בגודל הקלט. $\square$

19 שבוע 11 - 27.05.19

19.1 רדוקציות פולינומיות

על מנת לסקור בצורה יותר רצינית את המחלקה NP אנו נוקטים בגישה דומה לגישה שבה נקטנו כאשר עסקנו בתורת החישוביות – רדוקציות. הנה ההגדרות הרלוונטיות:

הגדרה 19.1 (פונקציה ניתנת לחישוב בזמן פולינומי)**.** $f:\Sigma^*\to\Sigma^*$ היא **פונקציה ניתנת לחישוב בזמן פולינומי** אס"ם קיימת מכונת טיורינג דטרמיניסטית שעל כל קלט $w\in\Sigma^*$ עוצרת בזמן פולינומיאלי ב-$|w|$ עם $f(w)$ על הסרט.

הגדרה 19.2 (רדוקציה פולינומית)**.** בהינתן שתי שפות $A,B\subseteq\Sigma^*$ נאמר ש-A ניתנת **לרדוקציה פולינומית** ל-B, ונסמן $A\le_p B$, אם קיימת פונקציה $f:\Sigma^*\to\Sigma^*$ שניתנת לחישוב בזמן פולינומי, כך שלכל מילה $w\in\Sigma^*$ מתקיים:

$$w\in A\leftrightarrow f(w)\in B$$

משפט 19.3 (משפט הרדוקציה - לרדוקציות פולינומיות)**.** יהיו $A,B\subseteq\Sigma^*$ שתי שפות. אם $A\le_p B$ וגם $B\in\mathsf{P}$ אז $A\in\mathsf{P}$.

הוכחה. היות ש-$A\le_p B$ אנו יודעים שקיימת פונקציה ניתנת לחישוב בזמן פולינומי f כך שלכל מילה $w\in\Sigma^*$ מתקיים $w\in A\leftrightarrow f(w)\in B$. היות ש-$f$ פונקציה ניתנת לחישוב בזמן פולינומי, קיימת מכונת טיורינג דטרמיניסטית M_f שבהינתן קלט w מחשבת את $f(w)$ בזמן פולינומיאלי. נתון גם כי $B\in\mathsf{P}$ ולכן קיימת מכונת טיורינג דטרמיניסטית M_B שמכריעה את B בזמן פולינומיאלי. כדי להוכיח ש-$A\in\mathsf{P}$ אנו נציג מכונת טיורינג דטרמיניסטית M_A שמכריעה את A בזמן פולינומיאלי:

בהינתן קלט $w\in\Sigma^*$, המכונה M_A תפעיל את M_f על מילת הקלט כדי לחשב את $f(w)$.

לאחר מכן, המכונה תריץ את M_B על הקלט $f(w)$ ותענה כמוה.

שימו לב שמדובר בדיוק באותה מכונה שבנינו במשפט הרדוקציה לרדוקציות מיפוי (משפט 14.6), לכן $L(M_A)=A$ וגם M_A עוצרת על כל קלט. נותר להצדיק מדוע זמן הריצה של M_A פולינומי ביחס לקלט $|w|$: חישוב $f(w)$ לוקח זמן פולינומיאלי ב-$|w|$ (בגלל התנאי על המכונה M_f); בפרט, זה אומר ש-$|f(w)|$ פולינומיאלי ב-$|w|$ (אחרת, M_f לא הייתה פולינומית באורך הקלט שלה). הרצת M_B על הקלט $|f(w)|$ לוקח זמן פולינומי ב-$|f(w)|$. מכאן נובע ש-M_A רצה בזמן פולינומיאלי ב-$|w|$ היות שפולינומים סגורים תחת הרכבה. $\square$

19.2 בעיית הספיקות

בעיית הספיקות (satisfiability problem) היא בעיית הכרעה שבהינתן נוסחה בוליאנית קובעת האם קיימת השמה מספקת (השמה שתיתן ערך 1) למשתני הנוסחה. השפה שנתעניין בה היא:

$$\mathsf{SAT}:=\{\langle\varphi\rangle:\text{נוסחה בוליאנית ספיקה } \varphi\}$$

נוסחה בוליאנית מעל קבוצת משתנים $X=\{x_1,\dots,x_n\}$ היא מחרוזת שנגזרת מדקדוק חסר ההקשר הבא:

$$B\to 0|1|x_1|\dots|x_n|\neg B|B\vee B|B\wedge B$$

כאשר הסימן $\neg$ מייצג פעולת שלילה (not), הסימן $\vee$ מייצג פעולת איווי (or) והסימן $\wedge$ מייצג פעולת גימום (and).
נוסחה בוליאנית φ תיקרא **ספיקה** $\iff$ קיימת השמה $f:X\to\{0,1\}$ כך ש-φ מקבלת ערך 1 תחת ההשמה f.
ליטרל הוא משתנה או שלילתו. למשל, אם x_1 ו-x_2 הם משתנים אז $\neg x_1$ ו-x_2 הם ליטרלים.
פסוקית היא דיסיוניקציה מרובה של ליטרלים – נוסחה שמורכבת ממספר ליטרלים שמחוברים ע"י פעולת $\vee$. למשל, הנוסחה $\neg x_1\vee x_2$ תיקרא פסוקית.
צורה נורמלית קוניוקטיבית (conjunctive normal form) או בקיצור CNF, היא נוסחה בוליאנית שמורכבת מקוניוקציה מרובה של פסוקיות. באופן סכמתי, נוסחה בצורה נורמלית קוניוקטיבית נראית כך:

$$\left(\ell_1^{(1)}\vee\ell_2^{(1)}\vee\dots\vee\ell_{c_1}^{(1)}\right)\wedge\left(\ell_1^{(2)}\vee\ell_2^{(2)}\vee\dots\vee\ell_{c_2}^{(2)}\right)\wedge\dots\wedge\left(\ell_1^{(m)}\vee\ell_2^{(m)}\vee\dots\vee\ell_{c_m}^{(m)}\right)$$

כאשר $\ell_j^{(i)}$ הוא ליטרל לכל $i\in[m]$ ו-$j\in[c_i]$; יש m פסוקיות, ובפסוקית ה-i יש c_i ליטרלים.
נאמר על נוסחה בוליאנית שהיא **בצורה נורמלית 3CNF** אס"ם בכל פסוקית יש 3 ליטרלים. וריאציה נוספת של בעיית ה-SAT היא בעיית ה-3SAT שמוגדרת כך:

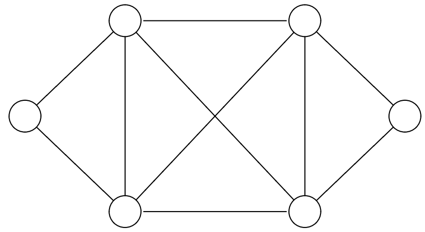
$$\mathsf{3SAT}:=\{\langle\varphi\rangle:\text{נוסחה בוליאנית ב-3CNF ספיקה } \varphi\}$$

טענה 19.4. $\mathsf{SAT}\in\mathsf{NP}$ וגם $\mathsf{3SAT}\in\mathsf{NP}$.

הוכחה. שתי ההוכחות מאוד דומות; מכונת טיורינג לא-דטרמיניסטית N עבור SAT תפעל כך: בהינתן נוסחה בוליאנית φ, המכונה N תנחש השמה למשתנים בנוסחה; לאחר מכן N תבדוק האם ההשמה מספקת את φ (בדיקה שניתן לבצע בזמן פולינומיאלי). $\square$

19.3 בעיית הקליקה

בהינתן גרף לא-מכוון $G = \langle V, E\rangle$, **קליקה** S ב-G היא תת-קבוצה של קודקודים $S \subseteq V$ כך שלכל $u_1 \neq u_2 \in S$ מתקיים $\{u_1, u_2\} \in E$. במילים: קיימת צלע בין כל זוג קודקודים ב-S. לדוגמה, בגרף הבא יש קליקה בגודל 4 (תוכלו למצוא אותה?):



נגדיר את השפה הבאה:

$$\text{CLIQUE} := \left\{\langle G, k\rangle : \substack{G \text{ גרף לא-מכוון} \\ \text{קיימת ב-}G\text{ קליקה בגודל } k}\right\}$$

טענה 19.5. CLIQUE $\in$ NP.

הוכחה. נשתמש באיפיון שהוכחנו במשפט 18.3. נבנה מוודא פולינומיאלי T עבור CLIQUE. המוודא יקבל כקלט $\langle G, k\rangle \# S$ כאשר S היא תת-קבוצה של קודקודי G בגודל k. המוודא יבדוק ש-S היא קליקה (כלומר, יש צלע המחברת בין כל שני קודקודים מ-S) ויקבל אם"ם S היא קליקה. בדיקה זו יכולה להתבצע בזמן פולינומיאלי כי יש לכל היותר $\binom{|V|}{2} = O\left(|V|^2\right)$ זוגות של קודקודים לבדוק. □

19.4 רדוקציה פולינומית מ-3SAT ל-CLIQUE

משפט 19.6. 3SAT $\leq_p$ CLIQUE.

הוכחה. כדי להוכיח את הטענה, עלינו להראות פונקציה ניתנת לחישוב בזמן פולינומיאלי f שבהינתן קלט $\langle\varphi\rangle$ לבעיית ה-3SAT מוציאה קלט $\langle G, k\rangle$ לבעיית ה-CLIQUE, כך שמתקיים:

$$\langle\varphi\rangle \in \text{3SAT} \leftrightarrow \langle G, k\rangle \in \text{CLIQUE}$$

<u>הבניה:</u> נסמן את φ בתור

$$\varphi = \left(\ell_1^{(1)} \vee \ell_2^{(1)} \vee \ell_3^{(1)}\right) \wedge \left(\ell_1^{(2)} \vee \ell_2^{(2)} \vee \ell_3^{(2)}\right) \wedge \cdots \wedge \left(\ell_1^{(m)} \vee \ell_2^{(m)} \vee \ell_3^{(m)}\right)$$

בפרט מספר הפסוקיות ב-φ הוא m. קבוצת הקודקודים V של G יהיו הליטרלים של φ (אותו ליטרל בשתי פסוקיות שונות מגדיר שני קודקודים שונים; בפרט $|V| = 3m$):

$$V := \left\{\ell_j^{(i)} \mid i \in [m], j \in [3]\right\}$$

נגדיר את הקבוצות:

$$E_s = \{\{u_1, u_2\} \mid u_1, u_2 \in V, u_1 \text{ and } u_2 \text{ in the same clause}\}$$
$$E_c = \{\{u_1, u_2\} \mid u_1, u_2 \in V, u_1 \text{ and } u_2 \text{ are identified with a variable and its negation}\}$$

הצלעות של G יהיו:

$$E := V \times V \setminus (E_s \cup E_c)$$

בנוסף נגדיר $k := m$ ונחזיר את הזוג $\langle G, k\rangle$, כאשר $G = \langle V, E\rangle$.

<u>הרדוקציה פולינומיאלית</u> כי $|V| = 3m$ ו-$|E|$ פולינומיאלי ב-$|V|$. בנוסף, החישוב של הקבוצות E_s ו-E_c יכול להתבצע בזמן פולינומיאלי ב-$|V|$.

<u>נכונות הרדוקציה:</u>

- נניח כי $\langle\varphi\rangle \in$ 3SAT ונוכיח $\langle G, k\rangle \in$ CLIQUE. היות ש- $\langle\varphi\rangle \in$ 3SAT קיימת השמה מספקת $g : X \to \{0, 1\}$ ל-φ. מכאן נובע שבכל פסוקית $i \in [m]$ קיים $j \in [3]$ כך ש- $\ell_j^{(i)}$ מקבל ערך אמת 1 תחת ההשמה g. נגדיר קבוצה S של ליטרלים שמכילה ליטרל מסתפק אחד מכל פסוקית ב-φ. עלינו להראות ש-S היא קליקה ב-G בגודל k. הגודל ברור כי $k = m$ ו-$|S| = m$. יהיו $u_1 \neq u_2 \in S$, נרצה להראות כי $\{u_1, u_2\} \in E$. אכן, נשים לב ש-u_1 ו-u_2 הם ליטרלים מפסוקיות שונות ולא ייתכן שהם מזוהים עם משתנה ושלילתו (אחרת, תחת ההשמה g אחד מהליטרלים היה מקבל ערך 0 ולא היה בקבוצה S). לפיכך, $\{u_1, u_2\} \notin E_s$ וגם $\{u_1, u_2\} \notin E_c$ ולכן $\{u_1, u_2\} \in E$, כנדרש.

- נניח כי $\langle G,k\rangle \in$ CLIQUE ונוכיח $\langle\varphi\rangle \in$ 3SAT. כיוון ש-$\langle G,k\rangle \in$ CLIQUE אנו יודעים שקיימת קליקה S ב-G בגודל $k=m$. מהגדרת E, יש ב-S לכל היותר קודקוד אחד מכל פסוקית; מכיוון ש-$k=m$ ו-m מספר הפסוקיות ב-φ, אנו מקבלים שב-S יש בדיוק קודקוד אחד מכל פסוקית. אנו רוצים להראות שקיימת השמה מספקת ל-φ, ולכן נגדיר את ההשמה $g: X\to\{0,1\}$ באופן הבא:

$$g(x)=\begin{cases}1 & \text{The vertex } x \text{ is in } S\\ 0 & \text{Otherwise}\end{cases}$$

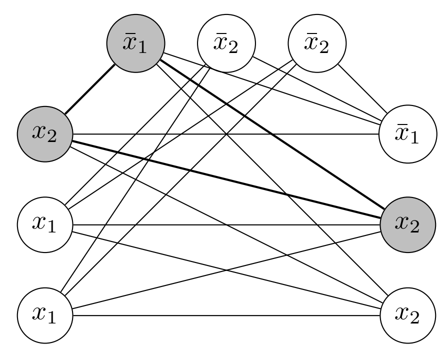
נטען כי g היא השמה מספקת של φ: מכל פסוקית של φ יש בדיוק קודקוד אחד ב-S. נשים לב שעבור קודקוד מסוים ℓ ב-S שמזוהה עם המשתנה x אנו למעשה מגדירים $g(x)=\begin{cases}1 & \ell=x\\ 0 & \ell=\neg x\end{cases}$, כי לא ייתכנו שני קודקודים ב-S שמזוהים עם משתנה ושלילתו. לכן כל קודקוד מ-S מקבל תחת ההשמה g ערך אמת $\Leftarrow$ כל פסוקית ב-φ מקבלת ערך אמת תחת ההשמה g $\Leftarrow$ הפסוקית φ מסתפקת תחת ההשמה g.

□

דוגמה 19.7. נתבונן בנוסחה φ הבאה:

$$\varphi=(x_1\vee x_1\vee x_2)\wedge(\neg x_1\vee\neg x_2\vee\neg x_2)\wedge(\neg x_1\vee x_2\vee x_2)$$

נוסחה זו ספיקה עבור ההשמה $x_1\mapsto 0$ ו- $x_2\mapsto 1$. הגרף שנבנה ע"י הרדוקציה הוא:



איור 19.1: הגרף שנבנה ע"י הרדוקציה במשפט 19.6. באיור מודגשת קליקה בגודל 3 בגרף.

19.5 שלמות ב-NP

הגדרה 19.8 (שלמות ב-NP)**.** שפה L תיקרא NP**-שלמה (NP-complete)** אם"ם מתקיים:

- $L\in$ NP.
- L היא NP**-קשה (NP-hard)**: לכל שפה $L'\in$ NP מתקיים $L'\le_p L$.

הערה 19.9. מספר הערות בנוגע למושגים NP-קשה ו-NP-שלם והקשר לבעיה $\mathsf{P}\overset{?}{=}\mathsf{NP}$:

- נסמן ב-NPH את מחלקת השפות שהן NP-קשות; נסמן ב-NPC את מחלקת השפות שהן NP-שלמות.
- אם L היא NP-שלמה ו-$L\in$ P אז P $=$ NP. וודאו שמובן לכם שזה נובע מההגדרות וממשפט הרדוקציה לרדוקציות פולינומיות.
- אם A היא NP-קשה וגם $B\le_p A$ אז B גם NP-קשה. הסיבה לכך היא שרדוקציות פולינומיות הן טרנזיטיביות – כיוון ש-$A\in$ NPH מתקיים לכל שפה $C\in$ NP כי $C\le_p A$; בשילוב עם הנתון $B\le_p A$ נובע $C\le_p B$. מכאן ש-$B\in$ NPH.

לפי שעה, לא ברור שבכלל קיימות שפות שהן NP-שלמות. בהרצאה הבאה נוכיח את משפט קוק-לוין אשר קובע: יש שפות NP-שלמות.

20 שבוע 11 - 29.05.19

20.1 משפט קוק-לוין

משפט 20.1 (משפט קוק-לוין)**.** השפה SAT היא NP-שלמה. (לשון אחר: $\mathsf{SAT} \in \mathsf{P}$ אם"ם $\mathsf{P} = \mathsf{NP}$.)

הוכחה. בטענה 19.4 ראינו ש-$\mathsf{SAT} \in \mathsf{NP}$. נותר להראות ש-SAT היא NP-קשה; לשם כך, נראה שלכל שפה $L \in \mathsf{NP}$ מתקיים $\mathsf{SAT}\ L \leq_p$. נקבע שפה $L \in \mathsf{NP}$ ונציג פונקציה ניתנת לחישוב בזמן פולינומיאלי f_L שבהינתן קלט w ל-L, f_L תחזיר נוסחה בוליאנית φ, כך שמתקיים:

$$w \in L \leftrightarrow \langle\varphi\rangle \in \mathsf{SAT}$$

אנו ניעזר בנתון היחיד שיש לנו על L – ידוע כי $L \in \mathsf{NP}$. לכן, קיימת מכונת טיורינג אי-דטרמיניסטית M שמכריעה את L בזמן פולינומיאלי. תהי $t(n)$ פונקציית הסיבוכיות של M.
הרעיון של הרדוקציה הוא בהינתן מילה w לבנות נוסחה φ שמבטאת "יש חישוב מקבל של M באורך לכל היותר $t(|w|)$ על w".
בהמשך נקבע $n = |w|$.
באופן יותר קונקרטי, φ כנ"ל תבטא את הקיום של סדרת קונפיגורציות $C_0, C_1, \ldots, C_m$ (עבור $m \leq t(n)$) כך שמתקיים:

- C_0 קונפיגורציה התחלתית של M על w.
- C_{i+1} עוקבת ל-C_i לכל $0 \leq i < m$.
- C_m קונפיגורציה מקבלת.

נשים לב שבריצה של M על w, מספר התאים הרלוונטים בסרט (התאים שיש סיכוי שנגיע אליהם) הוא לכל היותר $t(n)$. כך גם מספר הקונפיגורציות עד לעצירה קטן שווה מ-$t(n)$.
אנו נייצג ריצה של M על w בתור מטריצה בגודל $(t(n)+3) \times t(n)$, כאשר בכל כניסה במטריצה מצויה אות מתוך $C := Q \cup \Gamma \cup \{\#\}$, וקונפיגורציה מיוצגת ע"י מילה ב-$\#\Gamma^* Q \Gamma^* \#$.

#								#
#	q_0	w_1	$\ldots$	w_n	_	$\ldots$	_	#

הנוסחה φ תורכב כך:

$$\varphi = \varphi_{cell} \wedge \varphi_{init} \wedge \varphi_{acc} \wedge \varphi_{move}$$

כאשר:

- φ_{cell} - דואגת שבכל כניסה במטריצה יש אות אחת בדיוק מ-C.
- φ_{init} - דואגת שבקומה הראשונה במטריצה יש את הקונפיגורציה ההתחלתית של M על w.
- φ_{acc} - דואגת שקיימת קונפיגורציה מקבלת.
- φ_{move} - דואגת שטיפוס בין קומות מתאים למעבר בין קונפיגורציות עוקבות.

נגדיר את המשתנים כך:

$$X := \{x_{i,j,s} \mid i \in [t(n)+3], j \in [t(n)], s \in C\}$$

הרעיון המרכזי הוא לארגן את הנוסחה φ כך שבהשמה $f : X \to \{0,1\}$, אם $f(x_{i,j,s}) = 1$ אז בכתובת ה-i,j יש את האות s. למשל, אם $f(x_{3,2,\mathtt{a}}) = 1$ אז:

2	a
	3

הנוסחה φ_{init} מוגדרת כך:

$$\varphi_{init} := x_{1,1,\#} \wedge x_{2,1,q_0} \wedge x_{3,1,w_1} \wedge \cdots \wedge x_{n+2,1,w_n} \wedge x_{n+3,1,\_} \wedge \cdots \wedge x_{t(n)+2,1,\_} \wedge x_{t(n)+3,1,\#}$$

הנוסחה φ_{cell} מוגדרת כך:

$$\varphi_{cell} \coloneqq \bigwedge_{\substack{i\in[t(n)+3]\\ j\in[t(n)]}} \left(\left(\bigvee_{s\in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s_1\neq s_2\in C} \left(\neg x_{i,j,s_1} \vee \neg x_{i,j,s_2} \right) \right) \right)$$

החלק $\left(\bigvee_{s\in C} x_{i,j,s}\right)$ בנוסחה אומר "יש לפחות אות אחת בכל תא"; החלק $\bigwedge_{s_1\neq s_2\in C} \left(\neg x_{i,j,s_1} \vee \neg x_{i,j,s_2}\right)$ אומר "אין שתי אותיות שונות באותו תא".
הנוסחה φ_{acc} מוגדרת כך:

$$\varphi_{acc} \coloneqq \bigvee_{\substack{i\in[t(n)+3]\\ j\in[t(n)]}} x_{i,j,q_{acc}}$$

כלומר, קיימת כתובת שמאכלסת את q_{acc}.
הרעיון מאחורי φ_{move} הוא שניתן להבטיח עדכון חוקי של קונפיגורציות על ידי דרישה על "חלונות" בגודל 2×3:

$j+1$	s_4	s_5	s_6
j	s_1	s_2	s_3
	i	$i+1$	$i+2$

כאשר $s_i \in C$ לכל $i \in [6]$. נגדיר את φ_{move} ע"י:

$$\varphi_{move} \coloneqq \bigwedge_{\substack{i\in[t(n)+1]\\ j\in[t(n)-1]}} \textsf{legal}\left(i,j\right)$$

כאשר:

$$\textsf{legal}\left(i,j\right) \coloneqq \bigvee_{(s_1,s_2,\ldots,s_6)\in W} \left(x_{i,j,s_1} \wedge x_{i+1,j,s_2} \wedge x_{i+2,j,s_3} \wedge x_{i,j+1,s_4} \wedge x_{i+1,j+1,s_5} \wedge x_{i+2,j+1,s_6} \right)$$

הקבוצה W תהיה קבוצת החלונות החוקיים והיא מורכבת מ-4 סוגים של חלונות:

- לכל $c,d,e\in\Gamma$ נרשה חלון

c	d	e
c	d	e

- לכל מעבר $(q_2,b,\mathrm{R}) \in \delta\left(q_1,a\right)$ נרשה את החלונות הבאים:

b	q_2	c
q_1	a	c

c	b	q_2
c	q_1	a

d	c	b
d	c	q_1

q_2	c	d
a	c	d

a	c	d
a	c	d

- לכל מעבר $(q_2,b,\mathrm{L}) \in \delta\left(q_1,a\right)$ נרשה את החלונות הבאים:

q_2	c	b
c	q_1	a

d	q_2	c
d	c	q_1

e	d	q_2
e	d	c

c	b	d
q_1	a	d

b	c	d
a	c	d

- באופן דומה נטפל בחלונות עבור # (הסמל # משמש כמחוון לקצה הסרט).

נשים לב שיש מספר פולינומיאלי של חלונות חוקיים.
נותר להוכיח ש-$w \in L$ אס"ם φ ספיקה. אם $w \in L$ אז קיימת ריצה מקבלת בזמן פולינומיאלי של M על w, ריצה זו משרה השמה מספקת עבור φ. בכיוון ההפוך, אם יש השמה מספקת עבור φ אז השמה זו משרה ריצה מקבלת של M על w. השלמת הפרטים הושארה כתרגיל. $\square$

21 שבוע 12 - 05.06.19

21.1 רדוקציה פולינומית נוספת מ-3SAT ל-CLIQUE

במשפט 19.6 ראינו רדוקציה פולינומית מ-3SAT ל-CLIQUE. לאור משפט קוק-לוין (משפט 20.1) והרדוקציה מ-CNFSAT (הוגדרה בתרגול) ל-3SAT, אנו מקבלים ש-CLIQUE היא NP-שלמה.
נרצה להראות רדוקציה פולינומית נוספת מ-3SAT ל-CLIQUE כדי לעבות את ארגז הכלים שלנו.
בהינתן נוסחה φ בצורת 3CNF נרצה להחזיר (G, k) כאשר G גרף לא-מכוון ו-k מספר טבעי, כך ש-φ ספיקה אם"ם קיימת קליקה בגודל k בגרף G.
נכתוב את φ ע"י:

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

כאשר $C_i := \ell_1^{(i)} \vee \ell_2^{(i)} \vee \ell_3^{(i)}$ ו-$\ell_j^{(i)}$ הוא ליטרל מעל המשתנים $\{x_1, \ldots, x_n\}$, לכל $j \in [3]$ ו-$i \in [m]$.
נבחין כי מתוך $2^3 = 8$ השמות אפשריות לפסוקית, רק השמה אחת אינה מספקת (ההשמה שנותנת לכל הליטרלים ערך שקר); מכאן שלכל פסוקית יש 7 השמות מספקות.
נגדיר את הגרף G:

$$V = \left\{ f_1^{(i)}, \ldots, f_7^{(i)} \mid i \in [m] \right\}$$

כאשר $f_j^{(i)}$ מייצג השמה מספקת למשתני פסוקית C_i.
לדוגמה, עבור הפסוקית $C = x_1 \vee \overline{x_2} \vee \overline{x_3}$ אנו נייצר את הקודקודים הבאים:

000 001 010 100

101 110 111

שימו לב שלא כללנו את הקודקוד 011. הסדר של ההשמה בתוך הקודקוד מתאים לסדר הופעת המשתנים בפסוקית.
את צלעות הגרף נגדיר כך:

$$E = \left\{ \left\{ f_{j_1}^{(i_1)}, f_{j_2}^{(i_2)} \right\} \mid f_{j_1}^{(i_1)} \text{ and } f_{j_2}^{(i_2)} \text{ agrees on the assignment of joint variables} \right\}$$

שימו לב שלא תיתכן צלע מהצורה $\left\{ f_{j_1}^{(i)}, f_{j_2}^{(i)} \right\}$.
בנוסף, נגדיר $k := m$ (מספר הפסוקיות).
אם φ ספיקה אז קיימת לה השמה מספקת. השמה זו מספקת כל פסוקית ולכן לכל פסוקית $i \in [m]$ קיים j_i כך ש-$f_{j_i}^{(i)}$ מסכים עם ההשמה. השמה זו משרה קליקה בגודל k בגרף. בכיוון ההפוך, אם יש קליקה בגודל k בגרף אז קיימת השמה שמספקת את כל הפסוקיות ב-φ. (השלמת הפרטים הושארה כתרגיל).

22 שבוע 13 - 10.06.19

22.1 בעיית הסכומים החלקיים

בעיה 22.1 (בעיית הסכומים החלקיים (subset sum problem)).
קלט:

- קבוצה $A = \{a_i\}_{i=1}^{k}$ כאשר לכל $i \in [k]$ מתקיים $a_i \in \mathbb{N}$.
- מספר יעד $s \in \mathbb{N}$.

פלט: האם קיימת תת-קבוצה $B \subseteq A$ כך ש- s = $\sum_{a \in B} a = s$.
בהזמנות זו נגדיר את הבעיה גם כשפה:

$$\text{SubsetSum} := \left\{ \langle A, s \rangle \mid {}^{A \subseteq \mathbb{N}, s \in \mathbb{N}}_{\exists B \subseteq A \text{ s.t. } \sum_{a \in B} a = s} \right\}$$

דוגמה 22.2. עבור $A = \{5, 2, 10, 4, 7\}$ ו-$s = 18$ יש פתרון לבעיית הסכומים החלקיים. ניקח $B = \{5, 2, 4, 7\} \subseteq A$ ונווכח כי מתקיים:

$$5 + 2 + 4 + 7 = 18$$

משפט 22.3. $\text{SubsetSum} \in \text{NPC}$.

הוכחה. נראה כי $\text{SubsetSum} \in \text{NP}$ וגם $\text{SubsetSum} \in \text{NPH}$.

1. $\text{SubsetSum} \in \text{NP}$: עד (witness) לשייכות ל-SubsetSum היא תת-הקבוצה B. אורך העד פולינומיאלי בגודל הקלט (שהרי $|B| \leq |A|$), וניתן לבדוק את התנאי $\sum_{a \in B} a \stackrel{?}{=} s$ בזמן פולינומיאלי.
2. $\text{SubsetSum} \in \text{NPH}$: נראה רדוקציה פולינומית מ-3SAT ל-SubsetSum. בהינתן נוסחה φ ב-3CNF נבנה $\langle A, s \rangle$ בזמן פולינומיאלי כך ש- φ ספיקה אם ורק אם $\langle A, s \rangle \in \text{SubsetSum}$.

   תהי φ נוסחה כזו בעלת m פסוקיות $C_1, \ldots, C_m$ מעל המשתנים $X = \{x_1, \ldots, x_n\}$. הקבוצה A תכיל $2n + 2m$ מספרים, כל אחד מהם בן $n + m$ ספרות (בבסיס עשרוני).

   את הבניה של A נוח לתאר באמצעות טבלה בעלת $n + m$ עמודות, כאשר כל עמודה מציינת "ספרה"; ו-$2n + 2m$ שורות, כאשר כל שורה מציינת מספר בקבוצה שלנו.

   - לכל משתנה x_i, $i \in [n]$, נחזיק שני מספרים f_i ו-t_i. נשתמש בסימון $p\left[j\right]$ כדי להתייחס לספרה ה-j-ית במספר הטבעי p (כאשר הוא מיוצג בבסיס עשרוני); $j = 1$ מתייחסת לספרת ה-MSB.
     נגדיר את המספרים t_i ו-f_i ע"י הגדרת כל אחת מהספרות שלהם, לכל $j \in [n + m]$:

$$t_i\left[j\right] := \begin{cases} j \in \left[n\right], j = i: & 1 \\ j \in \left[n\right], j \neq i: & 0 \\ j \in [n+1, n+m], x_i \in C_{j-n}: & 1 \\ j \in [n+1, n+m], x_i \notin C_{j-n}: & 0 \end{cases}$$

     באופן דומה:

$$f_i\left[j\right] := \begin{cases} j \in \left[n\right], j = i: & 1 \\ j \in \left[n\right], j \neq i: & 0 \\ j \in [n+1, n+m], \overline{x_i} \in C_{j-n}: & 1 \\ j \in [n+1, n+m], \overline{x_i} \notin C_{j-n}: & 0 \end{cases}$$

   - לכל פסוקית C_i, $i \in [m]$, נחזיק שני מספרים p_i ו-q_i. לכל $j \in [n + m]$ נגדיר:

$$p_i\left[j\right] = q_i\left[j\right] := \begin{cases} j \in \left[n\right]: & 0 \\ j \in [n+1, n+m], j - n = i: & 1 \\ j \in [n+1, n+m], j - n \neq i: & 0 \end{cases}$$

   - מספר היעד s יוגדר כך: $s := 1^n 3^m$.

הערות לגבי הבניה:

- המספרים בטבלה מיוצגים בבסיס עשרוני ובאופן שהגדרנו את המספרים אנחנו לא צריכים לחשוש מ-carry.
- הקבוצה A היא multi-set (קבוצה שבה מותרת חזרה של איברים).

נכונות הרדוקציה:

- נניח ש-φ ספיקה. תהי $g : X \to \{0,1\}$ השמה מספקת. נבנה את הקבוצה B:
  - לכל $i \in [n]$ נבדוק האם $g\left(x_i\right) = 1$ ובמקרה כזה נוסיף את t_i ל-B; אחרת, נוסיף את f_i ל-B.
  - לכל $j \in [m]$:
    - אם g מספקת 3 ליטרלים בפסוקית C_j אז לא נוסיף ל-B את p_j ו-q_j.
    - אם g מספקת 2 ליטרלים בפסוקית C_j אז נוסיף את p_j ל-B.
    - אם g מספקת ליטרל בודד בפסוקית C_j אז נוסיף את p_j ו-q_j ל-B.

בסה"כ הקבוצה B מורכבת מלכל היותר $n + 2m$ מספרים. נטען כי מתקיים $\sum_{a \in B} a = 1^n 3^m$. נסמן $d := \sum_{a \in B} a$. ב-n-הספרות הראשונות (משמאל) של d אנו נקבל אחדות, שהרי כל תוספת של מספרים מתוך $\{p_j, q_j\}_{j \in [m]}$ לא משפיעה על הסכום ב-n-הספרות הראשונות; בנוסף בחרנו מספר אחד מתוך $\{f_i, t_i\}$ לכל משתנה x_i. מהאופן שבו המספרים הללו הוגדרו, אנו מקבלים שאכן n הספרות הראשונות משמאל של d הן אחדות. כעת נתעניין ב-m-הספרות האחרונות של d. נשים לב שאם משתנה x_i מופיע בפסוקית ה-C_j והוא מספק אותה תחת ההשמה g אז אנו נוסיף ל-B את המספר t_i ובנוסף יהיה מופיע 1 בעמודה $n + j$ בטבלה. תוספת של המספרים p_j ו-q_j מתבצעת בתור "ריפוד", ונבחרה כך שהמספר 3 ייצא בסכום של העמודה ה-$n + j$. באופן דומה אם המשתנה $\overline{x_i}$ מופיע בפסוקית ה-C_j והוא ליטרל שמספק אותה תחת g. הואיל ו-g השמה מספקת של φ, לפחות ליטרל אחד מתוך הפסוקית ה-j מסתפק תחת ההשמה g, ובהתאמה עם הריפוד שנבחר אנו נקבל ש-m-הספרות האחרונות של d הם 3^m. מומלץ להתבונן באיור 22.1.

- נניח ש-$\langle A, s\rangle \in$ SubsetSum. אזי קיימת תת-קבוצה $B \subseteq A$ כך ש- $\sum_{a \in B} a = s$. נרצה להראות שבחירת המספרים ב-B משרה השמה מספקת של φ. כיוון ש-n-הספרות הראשונות ב-s הן אחדות, אנו יודעים שלכל $i \in [n]$ <u>בדיוק</u> אחד מהמספרים $\{f_i, t_i\}$ שייך ל-B (אם לא היו נבחרים מספרים כלל עבור המשתנה ה-i, או לחילופין, אם היו נבחרים שני המספרים, אז לא היינו מקבלים 1 בספרה ה-i משמאל). אם עבור $i \in [n]$ המספר t_i נבחר אז נגדיר $g\left(x_i\right) = 1$; אם נבחר המספר f_i נגדיר $g\left(x_i\right) = 0$. נותר להראות שההשמה g שמושרית על-ידי B היא אכן השמה מספקת של φ. הואיל וה"ריפוד" שמוצע ע"י p_j ו-q_j יכול רק לתרום 2 לסכום בעמודה ה-$n + j$, כדי להגיע לסכום של 3 בעמודה הזו אנו חייבים לבחור משתנה אחד שתרם לסכום בעמודה $n + j$. הליטרל שמתאים למשתנה הזה מסתפק על-ידי ההשמה שהוגדרה, ולכן הפסוקית ה-j מסתפקת ע"י ההשמה g. זה נכון לכל $j \in [m]$ ולכן g השמה מספקת.

$\square$

הנוסחה הבוליאנית הבאה מוגדרת מעל $n = 3$ משתנים ומורכבת מ-$m = 4$ פסוקיות:

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

הקבוצה A שנבנית ע"י הרדוקציה:

	1	2	3	4	5	6	7
t_1	1	0	0	1	0	0	1
f_1	1	0	0	0	1	1	0
t_2	0	1	0	1	0	1	0
f_2	0	1	0	0	1	0	1
t_3	0	0	1	1	1	0	1
f_3	0	0	1	0	0	1	0
p_1	0	0	0	1	0	0	0
q_1	0	0	0	1	0	0	0
p_2	0	0	0	0	1	0	0
q_2	0	0	0	0	1	0	0
p_3	0	0	0	0	0	1	0
q_3	0	0	0	0	0	1	0
p_4	0	0	0	0	0	0	1
q_4	0	0	0	0	0	0	1
s	1	1	1	3	3	3	3

(המספר s לא חלק מהקבוצה A, הוא נמצא בטבלה לשם נוחות).
עבור ההשמה המספקת $x_1 = 1$, $x_2 = 0$, $x_3 = 0$ אנו נבחר לקבוצה B את המספרים הבאים:

$$B = \{t_1, f_2, f_3, p_1, q_1, p_2, q_2, p_3, q_3, p_4\}$$

איור 22.1: דוגמה לנוסחה φ והקבוצה A שנבנית ע"י הרדוקציה

22.2 סיבוכיות זיכרון

עד עכשיו עסקנו בסיבוכיות זמן. פרמטר נוסף שיעניין אותנו באלגוריתם הוא כמות הזיכרון שהוא צורך. גם כאן המודל המתמטי שבו נעסוק הוא מכונת טיורינג; הנה ההגדרה הפורמלית:

הגדרה 22.4 (סיבוכיות זיכרון)**.** בהינתן מכונת טיורינג M חד-סרטית שעוצרת על כל קלט, **סיבוכיות הזיכרון (space complexity)** של M היא פונקציה $s : \mathbb{N} \to \mathbb{N}$ כך שעל כל קלט באורך n, הריצה של M על הקלט משתמשת ב-$s(n)$ תאים לכל היותר. נאמר כי M **רצה בשטח** $s(n)$.

הגדרה 22.5. עבור פונקציה $s : \mathbb{N} \to \mathbb{N}$ נגדיר את מחלקות סיבוכיות הזיכרון הבאות:

$$\mathsf{SPACE} := \left\{ L \mid \substack{\text{קיימת מכונת טיורינג דטרמיניסטית} \\ \text{המכריעה את } L \text{ בשטח } O(s(n))} \right\}$$

$$\mathsf{NSPACE} := \left\{ L \mid \substack{\text{קיימת מכונת טיורינג אי-דטרמיניסטית} \\ \text{המכריעה את } L \text{ בשטח } O(s(n))} \right\}$$

כעת נעמוד על הקשרים הבסיסיים בין מחלקות סיבוכיות הזיכרון למחלקות סיבוכיות הזמן.

משפט 22.6. לכל פונקציה $f : \mathbb{N} \to \mathbb{N}$ מתקיים:

$$\mathsf{TIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$$

הוכחה. מכונת טיורינג שרצה בזמן $f(n)$ יכולה להשתמש בלכל היותר $f(n)$ תאים. $\square$

משפט 22.7. $\mathsf{SPACE}(f(n)) \subseteq \mathsf{TIME}\left(2^{O(f(n))}\right)$.

הוכחה. תהי M מכונת טיורינג דטרמיניסטית שרצה בשטח $f(n)$. קונפיגורציה של המכונה M היא איבר בקבוצה $Q \times \Gamma^{f(n)} \times [f(n)]$, וגודל הקבוצה הוא $|Q| f(n) 2^{O(f(n))} = 2^{O(f(n))}$ (שימו לב כי $f(n) = 2^{\log_2(f(n))}$ ו-$|Q|$ קבוע). M עוצרת על כל קלט ולכן לא חוזרת על אותה קונפיגורציה במהלך ריצתה. לפיכך, M עוצרת תוך $2^{O(f(n))}$ צעדים לכל היותר $\Leftarrow$ השפה שהכריעה נמצאת ב-$\mathsf{TIME}\left(2^{O(f(n))}\right)$. $\square$

הגדרה 22.8. נגדיר את מחלקות סיבוכיות הזיכרון הבאות:

$$\mathsf{PSPACE} \coloneqq \bigcup_{k=1}^{\infty} \mathsf{SPACE}\left(n^k\right)$$
$$\mathsf{NPSPACE} \coloneqq \bigcup_{k=1}^{\infty} \mathsf{NSPACE}\left(n^k\right)$$

מבחינת יחסי ההכלה בין המחלקות שונות, ידוע כי:

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXPTIME}$$

וגם $\mathsf{P} \neq \mathsf{EXPTIME}$. השוויון $\mathsf{PSPACE} = \mathsf{NPSPACE}$ נובע ממשפט סביץ' שנראה בהמשך.

דוגמה 22.9 (השפה SAT ב-PSPACE)**.** נטען כי SAT ניתנת להכרעה ב-PSPACE. הרעיון הוא לבנות מכונת טיורינג דטרמיניסטית שתעבור על כל ההשמות, ותקבל כאשר מצאה השמה מספקת. חשוב להבין שבזיכרון, בניגוד לזמן, ניתן לבצע שימוש חוזר. יהי $f_0, f_1, \ldots, f_{2^n-1}$ סידור של כל ההשמות מעל $n = |X|$ משתנים, כך שיש פונקציה חשיבה בשטח פולינומיאלי $g : \{0,1\}^n \to \{0,1\}^n \cup \{\perp\}$ המקיימת:

$$g\left(f_i\right) = \begin{cases} i < 2^n - 1: & f_{i+1} \\ i = 2^n - 1: & \perp \end{cases}$$

למשל, הסדר הלקסיקוגרפי משרה g כנ"ל. המכונה M תפעל על נוסחה φ כך:

1. M תכתוב על הסרט את f_0.
2. תהי f_i ההשמה על הסרט:
   (א) אם ההשמה על הסרט היא $\perp$ ← תדחה.
   (ב) M תשערך את φ לפי ההשמה f_i. אם φ מסתפקת ← תקבל.
   (ג) אחרת, תחשב את $g\left(f_i\right)$ באותם תאים בהם השתמשנו באיטרציות הקודמות.

מכאן נסיק, $\mathsf{SAT} \in \mathsf{PSPACE}$.

דוגמה 22.10 (השפה $\mathsf{ALL_{NFA}}$)**.** נתבונן בשפה

$$\mathsf{ALL_{NFA}} \coloneqq \left\{\langle A\rangle \mid L\left(A\right) = \Sigma^*\right\}$$

ניזכר שעבור אוטומטים דטרמיניסטיים בעיית האוניברסליות (לקבוע האם $L\left(A\right) = \Sigma^*$) כריעה ונמצאת ב-PTIME. לפיכך, גם השפה $\mathsf{ALL_{NFA}}$ כריעה. נתבונן בשפה המשלימה $\overline{\mathsf{ALL_{NFA}}}$:

$$\overline{\mathsf{ALL_{NFA}}} \coloneqq \left\{\langle A\rangle \mid L\left(A\right) \neq \Sigma^*\right\}$$

לא ניתן להכניס את $\overline{\mathsf{ALL_{NFA}}}$ ל-NP ע"י מוודא פולינומיאלי: יש סדרה של אוטומטים סופיים אי-דטרמיניסטיים $\left\{A_n\right\}_{n=1}^{\infty}$ כך ש-$L\left(A_n\right) \neq \Sigma^*$, הגודל של A_n פולינומי ב-n ואורך המילה הקצרה ביותר שלא מתקבלת ע"י A_n הוא אקספוננציאלי ב-n.
תיאור אפשרי של משפחה כזו (עם הוכחה) הועלה למודל (קישור).
בהרצאה הבאה נראה מכונת טיורינג אי-דטרמיניסטית שמכריעה את $\mathsf{ALL_{NFA}}$ בשטח לינארי.

23 שבוע 13 - 12.06.19

23.1 סיבוכיות זיכרון - המשך

נמשיך את דוגמה 22.10.

טענה 23.1. יהי NFA $A=\langle\Sigma,Q,Q_0,\delta,F\rangle$ שעבורו $L(A)\neq\Sigma^*$. אזי קיימת מילה $w\in\Sigma^*$ באורך לכל היותר $2^{|Q|}$ כך ש-$w\notin L(A)$.

הוכחה. מכיוון ש-$L(A)\neq\Sigma^*$ אנו מקבלים $L(\overline{A})\neq\emptyset$. האוטומט $\overline{A}$ הוא אוטומט סופי דטרמיניסטי אשר מתקבל ע״י ביצוע דטרמיניזציה של A ודואליזציה של תנאי הקבלה. לפיכך,

$$\left|\overline{A}\right|\leq 2^{|A|}$$

כאשר הסימון $|A|$ מתייחס למספר המצבים באוטומט A. ניזכר שאם B הוא אוטומט סופי דטרמיניסטי ו-$L(B)\neq\Sigma^*$ אז יש מילה u שלא מתקבלת ע״י B ואורכה חסום ע״י מספר המצבים ב-B. זאת מן הטעם הפשוט: אם אורך המילה הכי קצרה u שלא מתקבלת ע״י B גדול ממספר המצבים ב-B, אז מעיקרון שובך היונים קיים מצב שחוזר על עצמו, ולכן יש חלק במילה שניתן להסיר (החלק שבין שני המופעים השונים של המצב שחוזר על עצמו) ועדיין לקבל מילה שלא תתקבל ע״י B. המילה הזו תהיה קצרה יותר מ-u, בסתירה לבחירת u כמילה הקצרה ביותר שלא מתקבלת ע״י B.
במקרה שלנו, אם $L(\overline{A})\neq\emptyset$ אז יש מילה $w\in L(\overline{A})$ שגודלה חסום ע״י מספר המצבים ב-$\overline{A}$. כלומר, $\overline{A}$ מקבל מילה w באורך לכל היותר $\left|\overline{A}\right|=2^{|Q|}$. לפיכך, קיימת $w\in\Sigma^*$ באורך לכל היותר $2^{|Q|}$ שמקיימת $w\notin L(A)$, כנדרש. □

טענה 23.2. יהי NFA $A=\langle\Sigma,Q,Q_0,\delta,F\rangle$ שעבורו $L(A)\neq\Sigma^*$. אזי קיימת סדרה

$$S_0,S_1,\ldots,S_k$$

עבור $k\leq 2^{|Q|}$, $S_i\subseteq Q$ לכל $i\in[k]$ וגם:

1. $S_0=Q_0$.
2. לכל $0\leq i<k$ קיים $\sigma_i\in\Sigma$ כך ש-
$$S_{i+1}=\delta^*(S_i,\sigma_i)$$
3. $S_k\cap F=\emptyset$.

הוכחה. מטענה 23.1 אנו יודעים שקיימת מילה $w\notin L(A)$ כך ש-

$$w=\sigma_0\sigma_1\ldots\sigma_{k-1}$$

עבור $k\leq 2^{|Q|}$, ו-$\sigma_i\in\Sigma$ לכל $0\leq i\leq k-1$. הסדרה $S_0,S_1,\ldots,S_k$ היא ריצה מקבלת של $\overline{A}$ על w (זכרו ש-$\overline{A}$ הוא אוטומט סופי דטרמיניסטי שהתקבל ע״י ביצוע דטרמיניזציה של A באמצעות ה-subset construction ולאחר מכן ביצוע דואליזציה של תנאי הקבלה). מכיוון שמדובר בריצה מקבלת של $\overline{A}$ אנו יודעים שמתקיים $S_0=Q_0$ וגם ש-S_k הוא מצב מקבל ב-$\overline{A}$ וזה נכון אם ורק אם $S_k\cap F=\emptyset$. בנוסף, שימו לב שפונקציית המעברים של $\overline{A}$ מתלכדת עם פונקציית המעברים המורחבת δ^* של A (כפי שראינו במשפט 3.9). □

הערה 23.3. שימו לב שגם הכיוון ההפוך של הטענה האחרונה נכון. כלומר, אם קיימת סדרה $S_0,\ldots,S_k$ כנ״ל אז $L(A)\neq\Sigma^*$.

עתה, נבנה מכונת טיורינג אי-דטרמיניסטית N שבהינתן NFA A פועלת כך:

1. המכונה תאפס מונה $(i=0)$ ותכתוב על הסרט את Q_0 $(S_0=Q_0)$.
2. כל עוד $i\leq 2^{|Q|}$:
   (א) אם $S_i\cap F=\emptyset$ המכונה עוצרת ומקבלת.
   (ב) אחרת, המכונה מנחשת באופן לא דטרמיניסטי אות $\sigma_i\in\Sigma$, וכותבת על הסרט את $S_{i+1}=\delta^*(S_i,\sigma_i)$.
   (ג) המכונה מקדמת את i ב-1.
3. אם $i>2^{|Q|}$ המכונה עוצרת ודוחה.

נבחין כי $L(N)=\overline{\mathsf{ALL}_{\mathsf{NFA}}}$: אם N מקבלת את $\langle A\rangle$ אז N הצליחה לנחש מילה $w=\sigma_0\ldots\sigma_{r-1}$, $r\leq 2^{|Q|}$, שעבורה קיימת סדרה $S_0,\ldots,S_r$ כבטענה 23.2. לפיכך, w לא מתקבלת ע״י $\langle A\rangle$, ולכן $\langle A\rangle\in\overline{\mathsf{ALL}_{\mathsf{NFA}}}$. בכיוון ההפוך, אם $\langle A\rangle\notin L(N)$ אז זה אומר שלא קיימת מילה w באורך לכל היותר $2^{|Q|}$ שלא מתקבלת ע״י A. לפי טענה 23.1 זה גורר $\langle A\rangle\notin\overline{\mathsf{ALL}_{\mathsf{NFA}}}$.
סיבוכיות זיכרון: בכל שלב המכונה N שומרת את המונה i, את הקבוצה $S_i\subseteq Q$ ומשתמשת בזיכרון חסום עבור העדכונים הנדרשים. כמות הזיכרון בשימוש הוא לינארי ב-$|\langle A\rangle|$.
מכאן אנו מסיקים $\overline{\mathsf{ALL}_{\mathsf{NFA}}}\in\mathsf{NPSPACE}$.

23.2 שלמות ב-PSPACE

הגדרה 23.4. נאמר ששפה L היא PSPACE-שלמה אס"ם:

1. $L \in \mathsf{PSPACE}$.
2. L היא PSPACE-קשה: לכל שפה $L' \in \mathsf{PSPACE}$ מתקיים $L' \leq_p L$.

משפט 23.5. אם L היא PSPACE-קשה ו-$L \in \mathsf{NP}$ אז $\mathsf{NP} = \mathsf{PSPACE}$.

הוכחה. כבר התוודענו לכך ש- $\mathsf{NP} \subseteq \mathsf{PSPACE}$. נוכיח את הכיוון השני. בהינתן שפה $L' \in \mathsf{PSPACE}$, אנו יודעים שיש רדוקציה פולינומית f מ-L' אל L. בנוסף, כיוון ש-$L \in \mathsf{NP}$ קיימת מכונת טיורינג אי-דטרמיניסטית N שמכריעה את L בזמן פולינומיאלי. נכריע את L' ע"י מכונת טיורינג אי-דטרמיניסטית בזמן פולינומיאלי כך:

1. בהינתן מילת קלט w, נחשב את $f(w)$.
2. נכריע באופן אי-דטרמיניסטי (באמצעות המכונה N) האם $f(w) \in L$, ונחזיר את התשובה שהתקבלה.

היות שהפונקציה f חשיבה בזמן פולינומיאלי, אנו מקבלים $L' \in \mathsf{NP}$. □

הערה 23.6. בהגדרה 23.4 דרשנו שתהיה רדוקציה בזמן פולינומי מ-L' ל-L. יכולנו גם להגדיר רדוקציה מ-L' ל-L שפועלת בשטח פולינומיאלי. המוטיבציה מאחורי ההגדרה שלנו היא משפט 23.5. הוכחת המשפט לא הייתה אפשרית אם היינו דורשים רדוקציה מ-L' ל-L שפועלת בשטח פולינומיאלי.

24 שבוע 14 - 17.06.19

בהרצאה זו נראה שהשפה $\overline{\mathsf{ALL_{NFA}}}$ היא PSPACE-שלמה. בהרצאה הקודמת הראינו כי $\overline{\mathsf{ALL_{NFA}}} \in \mathsf{NPSPACE}$. בהרצאה הנוכחית נראה כי $\overline{\mathsf{ALL_{NFA}}}$ היא PSPACE-קשה ונוכיח תוצאה הידועה בשם "משפט סביץ'" שממנה ינבע $\mathsf{NPSPACE} = \mathsf{PSPACE}$.

24.1 השפה $\overline{\mathsf{ALL_{NFA}}}$ היא PSPACE-קשה

משפט 24.1. $\overline{\mathsf{ALL_{NFA}}}$ היא PSPACE-קשה.

הוכחה. נקבע $L \in \mathsf{PSPACE}$. נראה כי מתקיים $L \leq_p \overline{\mathsf{ALL_{NFA}}}$, קרי, קיימת רדוקציה פולינומית מ-L ל-$\overline{\mathsf{ALL_{NFA}}}$. תהי M מ"ט דטרמיניסטית עם סיבוכיות זיכרון פולינומית $s(n)$ המכריעה את L. בהינתן מילה $w \in \Sigma_M^*$, נבנה בזמן פולינומיאלי אוטומט NFA A כך שיתקיים:

$$w \in L \iff \langle A \rangle \in \overline{\mathsf{ALL_{NFA}}}$$

הרעיון הוא לבנות (בזמן פולינומיאלי בגודל $|w|$) אוטומט סופי אי-דטרמיניסטי A כך שיתקיים $x \in L(A)$ אם"ם אחד מהשניים מתקיים:

1. x איננה קידוד של ריצה של M על w.
2. x קידוד של ריצה דוחה של M על w.

אם נצליח לבנות A כנ"ל, הרדוקציה תהיה נכונה: אם $w \in L$ אז קיימת מילה x שמהווה קידוד של ריצה מקבלת של M על w ולכן x לא תתקבל ע"י A. מכאן ש-$L(A) \neq \Sigma_A^*$ ולכן $\langle A \rangle \in \overline{\mathsf{ALL_{NFA}}}$. בכיוון ההפוך, אם $L(A) \neq \Sigma_A^*$ אז קיימת מילה x שלא מקיימת את דרישה מספר 1 וגם לא מקיימת את דרישה מספר 2, ולכן x היא למעשה קידוד חוקי של ריצה מקבלת של M על w (זכרו כי M דטרמיניסטית והיא מכריעה את L; לפיכך, אם הריצה של M על w לא דוחה, היא בהכרח מקבלת). לאור זאת, $w \in L(M)$, ולכן $w \in L$.

נתאר את בניית האוטומט A. הא"ב של A יוגדר כך $\Sigma_A := \Gamma_M \cup (Q_M \times \Gamma_M) \cup \{\#\}$. נגדיר גם $n := |w|$. נקודד ריצה מקבלת של M על w כך:

$$\#C_0\#C_1\#\ldots\#C_k\#$$

כאשר C_0 קונפיגורציה התחלתית; C_{i+1} עוקבת ל-C_i; C_k מקבלת. קונפיגורציה תיוצג כך:

$$\#\gamma_1\gamma_2\ldots\gamma_{i-1}(q,\gamma_i)\gamma_{i+1}\ldots\gamma_{s(n)}\#$$

כאשר $\gamma_j \in \Gamma_M$ לכל $j \in [s(n)]$. הקונפיגורציה ההתחלתית C_0 תקודד כך:

$$\#(q_0,w_1)w_2\ldots w_n\_\_\ \ldots\ \_\#$$

כאשר $w = w_1w_2\ldots w_n$ ו- $w_j \in \Gamma_M$ לכל $j \in [n]$.

בהינתן קלט $x \in \Sigma_A^*$ האוטומט A יורכב משני רכיבים: הרכיב הראשון מקבל את x לפי הקריטריון הראשון (x איננה קידוד של ריצה של M על w) והרכיב השני מקבל את x לפי הקריטריון השני (x קידוד של ריצה דוחה של M על w). האוטומט ינחש עם איזה רכיב להתחיל (זכרו ש-A אוטומט אי-דטרמיניסטי). הרכיב השני פשוט יחסית – A מקבל את x אם x מכיל אות ב-$\Gamma \times \{q_{rej}\}$. הרכיב הראשון מורכב קצת יותר. אם x איננה קידוד של ריצה של M על w אז ייתכנו שתי אפשרויות שלכל אחת מהן מתאים "תת-רכיב" ב-A:

- x לא מתחיל בקידוד של הקונפיגורציה ההתחלתית. בתת-רכיב זה A בודק האם x מתחיל ב-C_0, ואם לא, מקבל.
- קיים ב-x הפרה של מעבר חוקי בין קונפיגורציות עוקבות. יהיו

$$\#\sigma_1\ldots\sigma_{s(n)}\#\sigma'_1\ldots\sigma'_{s(n)}$$

שתי קונפיגורציות עוקבות ב-x, $\sigma_j, \sigma'_j \in \Sigma_A$. נגדיר גם $\# =: \sigma_0 = \sigma'_0 = \sigma_{s(n)+1}$. לכל חלון בגודל 3 $(\sigma_{i-1},\sigma_i,\sigma_{i+1}) \in \Sigma_A^3$, $i \in [s(n)]$, נסמן ב-$\mathsf{next}(\sigma_{i-1},\sigma_i,\sigma_{i+1})$ את האות שאנו מצפים לפגוש במקום של σ'_i (האות האמצעית בחלון המקביל בקונפיגורציה העוקבת). הפונקציה next תוגדר כך[2] ($\gamma \in \Gamma_M$, $q, q' \in Q_M$ ו-$\sigma \in \Sigma_A$):

– $\mathsf{next}(\gamma_{i-1},\gamma_i,\gamma_{i+1}) = \mathsf{next}(\#,\gamma_i,\gamma_{i+1}) = \mathsf{next}(\gamma_{i-1},\gamma_i,\#) = \gamma_i$

[2] ההגדרה של הפונקציה next לקוחה מקובץ ההשלמה שהועלה ע"י אורנה למודל.

– $$\text{next}\left(\left(q,\gamma_{i-1}\right),\gamma_i,\gamma_{i+1}\right)=\text{next}\left(\left(q,\gamma_{i-1}\right),\gamma_i,\#\right)=\begin{cases}\gamma_i & \text{if}\left(q,\gamma_{i-1}\right)\to\left(q',\gamma'_{i-1},\mathrm{L}\right)\\ \left(q',\gamma_i\right) & \text{if}\left(q,\gamma_{i-1}\right)\to\left(q',\gamma'_{i-1},\mathrm{R}\right)\end{cases}$$

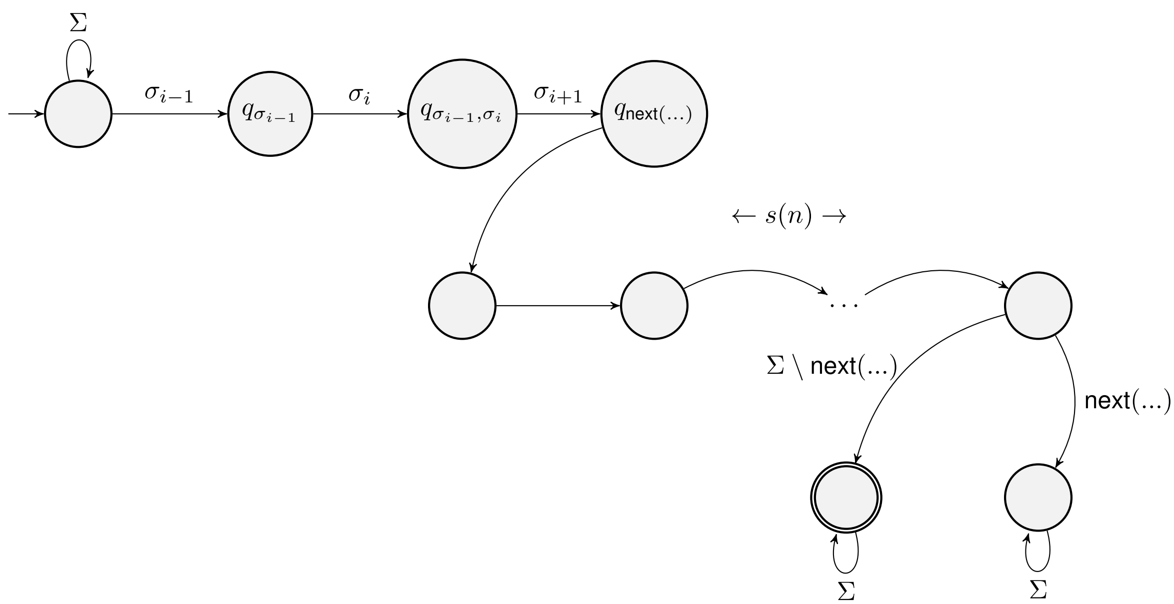
– $\text{next}\left(\gamma_{i-1},\left(q,\gamma_i\right),\gamma_{i+1}\right)=\text{next}\left(\#,\left(q,\gamma_i\right),\gamma_{i+1}\right)=\text{next}\left(\gamma_{i-1},\left(q,\gamma_i\right),\#\right)=\gamma'_i$ כאשר יש מעבר $\left(q,\gamma_i\right)\to\left(q',\gamma'_i,\Delta\right)$ ו- $\Delta\in\{\mathrm{L},\mathrm{R}\}$. בהנחה שהמכונה M לא נופלת מקצוות הסרט, המעברים $\left(q,\gamma_1\right)\to\left(q',\gamma'_1,\mathrm{L}\right)$ ו- $\left(q,\gamma_{s(n)}\right)\to\left(q',\gamma'_{s(n)},\mathrm{R}\right)$ לא אפשריים.

– $$\text{next}\left(\gamma_{i-1},\gamma_i,\left(q,\gamma_{i+1}\right)\right)=\text{next}\left(\#,\gamma_i,\left(q,\gamma_{i+1}\right)\right)=\begin{cases}\gamma_i & \text{if}\left(q,\gamma_{i+1}\right)\to\left(q',\gamma'_{i+1},\mathrm{R}\right)\\ \left(q',\gamma_i\right) & \text{if}\left(q,\gamma_{i+1}\right)\to\left(q',\gamma'_{i+1},\mathrm{L}\right)\end{cases}$$

– $\text{next}\left(\sigma_{s(n)},\#,\sigma'_1\right)=\#$.

באיור 24.1 מתואר באופן סכמתי כיצד A מנחש הפרה של קונפיגורציות עוקבות, בעזרת הפונקציה next.

נשים לב שהבניה של A פולינומיאלית כי בדיקה ש-x לא מתחיל בקידוד של הקופיגורציה ההתחלתית יכולה להיעשות באמצעות $O\left(s\left(n\right)\right)$ מצבים ; בדיקה האם קיימת הפרה של מעבר חוקי בין קונפיגורציות עוקבות יכולה להיעשות ב-$O\left(\left|\Sigma_A\right|^3\cdot s\left(n\right)\right)$ מצבים – שהרי יש $\left|\Sigma_A\right|^3$ חלונות בגודל 3 ולכל חלון כזה אנו בונים $O\left(s\left(n\right)\right)$ מצבים, בדומה לאיור 24.1. $\square$



איור 24.1 : איור סכמתי של "תת-רכיב" באוטומט A שמנחש הפרה של קונפיגורציות עוקבות.

24.2 משפט סביץ'

משפט סביץ' (Savitch's theorem), שהוכח ב-1970, הוא תוצאה חשובה הנוגעת לסיבוכיות מקום. המשפט מראה שבהינתן מכונת טיורינג אי-דטרמיניסטית שמכריעה שפה L עם סיבוכיות זיכרון $s\left(n\right)$ ניתן לבנות מכונת טיורינג דטרמיניסטית שמכריעה את אותה שפה L עם סיבוכיות זיכרון של $s\left(n\right)^2$. הנה הניסוח המדויק :

משפט 24.2 (משפט סביץ')**.** לכל פונקציית סיבוכיות זיכרון $s:\mathbb{N}\to\mathbb{N}$ חשיבה במקום כך ש- $s\left(n\right)\geq n$ מתקיים :

$$\text{NSPACE}\left(s\left(n\right)\right)\subseteq\text{SPACE}\left(s\left(n\right)^2\right)$$

הוכחה. בהינתן מכונת טיורינג אי-דטרמיניסטית M עם סיבוכיות זיכרון $s\left(n\right)$, נבנה מכונת טיורינג דטרמיניסטית שקולה M' עם סיבוכיות זיכרון $s\left(n\right)^2$.

סימונים והנחות :

- יש ל-M מצב התחלתי יחיד. אם יש ל-M מספר מצבים התחלתיים ניתן לבנות ממנה מכונת טיורינג שקולה בעלת מצב התחלתי יחיד. (זכרו כי M אי-דטרמיניסטית.)
- עבור מילה w נסמן ב-C_0^w את הקונפיגורציה ההתחלתית של M על w. היות שיש ל-M מצב התחלתי יחיד, גם הקונפיגורציה ההתחלתית היא יחידה.
- יש ל-M קונפיגורציה מקבלת יחידה, שנסמנה C_{acc}. באופן כללי ההנחה הזאת לא מתקיימת, אולם בקלות ניתן לשנות את M ולספק אותה. הרעיון הוא לשנות את M כך שבבואה לקבל מילה, M תמחק את הסרט, תזוז עם הראש הקורא שמאלה ורק אז תעבור למצב q_{acc}. במקרה כזה, הקונפיגורציה המקבלת תמיד תהיה $\ldots\ q_{acc}\_\_$.
- מספר הקונפיגורציות של M על w הוא לכל היותר $|Q_M| \cdot |\Gamma_M|^{s(n)} \cdot s(n) = 2^{O(s(n))}$. לפיכך קיים קבוע $d > 0$ כך שאין ל-M יותר מ-$2^{d \cdot s(n)}$ קונפיגורציות.

נבנה שגרה דטרמיניסטית $\text{reach}\left(C_1, C_2, t\right)$ המכריעה האם ניתן להגיע מקונפיגורציה C_1 לקונפיגורציה C_2 תוך לכל היותר t צעדים. סיבוכיות הזיכרון של $\text{reach}\left(C_1, C_2, t\right)$ תהיה $O\left(s\left(n\right) + \log t\right) \cdot \left(\log_2 t\right)$.
בהינתן מילה w, M' תריץ את $\text{reach}\left(C_0^w, C_{acc}, 2^{d \cdot s(n)}\right)$. נשים לב:

- M מקבלת את $w \iff C_{acc}$ ישיגה מ-C_0^w תוך $2^{d \cdot s(n)}$ צעדים $\iff M'$ מקבלת את w.
- סיבוכיות הזיכרון של M' היא:

$$O\left(s\left(n\right) + \log\left(2^{d \cdot s(n)}\right)\right) \cdot \log_2\left(2^{d \cdot s(n)}\right) = O\left(s\left(n\right) + d \cdot s\left(n\right)\right) \cdot d \cdot s\left(n\right) = O\left(s\left(n\right)^2\right)$$

כפי שרצינו.

השגרה reach תהיה שגרה רקורסיבית שתבדוק האם קיימת קונפיגורציה אמצעית C_{mid} כך ש-C_{mid} ישיגה מ-C_1 תוך $\left\lfloor \frac{t}{2} \right\rfloor$ צעדים ו-C_2 ישיגה מ-C_{mid} תוך $\left\lceil \frac{t}{2} \right\rceil$ צעדים. הנה התיאור המדויק:

1. (מקרה בסיס) אם $t = 1$ בדוק האם $C_1 = C_2$ או בדוק ש-C_2 עוקבת ל-C_1. אם כן $\leftarrow$ קבל; אם לא $\leftarrow$ דחה.
2. אם $t > 1$, אז לכל קונפיגורציה C_{mid} שמשתמשת ב-$s\left(n\right)$ תאים, בצע:
   (א) הרץ את $\text{reach}\left(C_1, C_{mid}, \left\lfloor \frac{t}{2} \right\rfloor\right)$.
   (ב) הרץ את $\text{reach}\left(C_{mid}, C_2, \left\lceil \frac{t}{2} \right\rceil\right)$.
   (ג) אם שתי ההרצות קיבלו $\leftarrow$ קבל.
3. דחה.

כדי לנתח את סיבוכיות הזיכרון של reach ניתן את דעתנו על גודל מחסנית הרקורסיה של reach. כאשר השגרה reach מפעילה את עצמה באופן רקורסיבי, היא שומרת במחסנית את הקונפיגורציות C_1 ו-C_2 ואת המספר t. לפיכך, בכל רמה ברקורסיה אנו משתמשים ב-$O\left(s\left(n\right) + \log t\right)$ זיכרון. עומק הרקורסיה הוא $\log_2 t$ כי בכל שלב אנו מחלקים את t ב-2. לפיכך, השגרה reach משתמשת ב-$O\left(s\left(n\right) + \log t\right) \cdot \left(\log_2 t\right)$ זיכרון. נעיר כי ניתן לחשב את הערך $s\left(n\right)$ ב-$O\left(s\left(n\right)\right)$ מקום כי $s\left(n\right)$ חשיבה במקום, לפי ההנחה. □

הערה 24.3. משפט סביץ' תקף גם אם מסירים את הדרישה ש-$s\left(n\right)$ חשיבה במקום. בנוסף, משפט סביץ' תקף גם עבור $s\left(n\right) \geq \log n$ (ראו "הערות בנוגע למשפט סביץ'" בהרצאה הבאה).

מסקנה מיידית מהמשפט:

מסקנה 24.4. $\text{PSPACE} = \text{NPSPACE}$.

25 שבוע 14 - 19.06.19

25.1 המחלקות L ו-NL

עד עכשיו דיברנו על פונקציות סיבוכיות זמן וזיכרון שהיו לכל הפחות לינאריות בגודל הקלט. כיום נדבר על סיבוכיות תת-לינארית (sub-linear). סיבוכיות זמן תת-לינארית לא תעניין אותנו משום שזמן תת-לינארי לא מאפשר לקרוא את הקלט במלואו. יחד עם זאת, סיבוכיות זיכרון תת-לינארית מאפשרת לקרוא את הקלט במלואו אבל תאפשר להשתמש "במעט" זיכרון. כאשר אנו מדברים על סיבוכיות זיכרון תת-לינארית אנו נשנה מעט את מודל החישוב שלנו.

מודל החישוב מודל החישוב הוא מכונת טיורינג עם שני סרטים:

- סרט הקלט - עליו כתובה מילת הקלט. ניתן לקרוא ממנו ולא ניתן לכתוב עליו (read only).
- סרט העבודה - משמש את מכונת טיורינג בחישוביה. ניתן לקרוא ממנו וגם ניתן לכתוב עליו (read/write).

סיבוכיות הזיכרון של מכונה כזו תימדד לפי כמות התאים שבהם השתמשה המכונה בסרט העבודה שלה.

הגדרה 25.1 (המחלקות L ו-NL)**.** L היא מחלקת השפות שכריעות בשטח לוגריתמי על מכונת טיורינג דו-סרטית דטרמיניסטית:

$$\mathsf{L}=\mathsf{SPACE}\left(\log n\right)$$

NL היא מחלקת השפות שכריעות בשטח לוגריתמי על מכונת טיורינג דו-סרטית אי-דטרמיניסטית:

$$\mathsf{NL}=\mathsf{NSPACE}\left(\log n\right)$$

דוגמה 25.2. נתבונן בשפה $\mathsf{EQ}=\left\{0^i1^i \mid i\geq 0\right\}$. בדוגמה 17.2 ראינו כי $\mathsf{EQ}\in\mathsf{TIME}\left(n\log n\right)$. כעת נראה ש- $\mathsf{EQ}\in\mathsf{L}$; כלומר, נראה מ"ט דטרמיניסטית דו-סרטית עם סיבוכיות זיכרון $O\left(\log n\right)$ כאשר n הוא אורך הקלט. המכונה מתחזקת בסרט העבודה שני מונים c_0 ו-c_1 אשר סופרים את מספר ה-0-ים וה-1-ים, בהתאמה. המכונה מקבלת אס"ם $c_0=c_1$. המכונה בבירור מכריעה את EQ. נשים לב כי $c_0+c_1=n$ כי מילת הקלט שייכת ל-$\{0,1\}^*$; בנוסף, $c_0,c_1\geq 0$, ולכן אנו מקבלים $c_0,c_1\leq n$. לפיכך, ניתן לייצג את c_0 ב-$\lfloor\log n\rfloor+1$ ביטים, וכנ"ל עבור c_1. מכאן ששמירת המונים צורכת שטח $O\left(\log n\right)$, ולכן סיבוכיות הזיכרון של המכונה היא $O\left(\log n\right)$, כנדרש.

דוגמה 25.3. נתבונן בבעיה הבאה:

$$\mathsf{PATH}:=\left\{\langle G,s,t\rangle \mid \text{קיים מסלול מ- } s \text{ ל- } t \text{ בגרף המכוון } G\right\}$$

נראה כי $\mathsf{PATH}\in\mathsf{NL}$. כלומר, יש מכונת טיורינג אי-דטרמיניסטית בעלת סיבוכיות זיכרון $O\left(\log n\right)$ ומכריעה את PATH. הרעיון הוא לגרום למכונה לבצע הדמיה של "טיול" על הגרף שמתחיל מהקודקוד s. אם במהלך הטיול הגענו לקודקוד t - נקבל; אחרת, נדחה. בכל שלב ננחש לאיזה קודקוד נתקדם. בגישה הזאת אנחנו יכולים לטייל על הגרף בלי סוף. כיוון שהמכונה מכריעה, היא חייבת לעצור על כל קלט. לשם כך אנו נתחזק מונה, ונעצור אם ערכו גדול מ-$|V|$, מספר הקודקודים. הנכונות מתבססת על העובדה הבאה: אם קיים מסלול בין s ל-t ב-G אז קיים מסלול בין s ל-t ב-G שאורכו לכל היותר $|V|$ (למשל, המסלול הקצר ביותר בין s ל-t מקיים זאת). כעת נראה כיצד אנחנו יכולים לממש "טיול" כזה עם שטח לוגריתמי. נתאר מכונת טיורינג אי-דטרמיניסטית M שמכריעה את PATH ופועלת כך:

1. M כותבת על סרט העבודה קודקוד נוכחי $v:=s$ ומונה $i:=0$.
2. עבור קודקוד נוכחי v ומונה i:
   (א) אם $v=t$: קבל.
   (ב) אם $i\geq|V|$: דחה.
   (ג) אחרת, M מנחשת קודקוד u שעוקב לקודקוד הנוכחי.
   (ד) M כותבת את u על הסרט, במקום v.
   (ה) M מגדילה את המונה i ב-1.

אכן $L\left(M\right)=\mathsf{PATH}$ שהרי קיים מסלול מ-s ל-t ב-G אס"ם קיימת ריצה של M שמטיילת על הגרף ומגיעה ל-t. בנוסף, את המונה i נוכל לייצג ב-$O\left(\log|V|\right)$ ביטים; גם כל קודקוד ניתן לייצג ב-$O\left(\log|V|\right)$ ביטים, שהרי יש $|V|$ קודקודים בסך-הכל. בסה"כ סיבוכיות הזיכרון לוגריתמית ביחס לגודל הקלט, ולכן $\mathsf{PATH}\in\mathsf{NL}$.

הערות בנוגע למשפט סביץ'

- במשפט סביץ' (משפט 24.2) דרשנו שפונקציית הסיבוכיות s תקיים $n \geq s(n)$. המשפט נכון גם אם $s(n) \geq \log n$. זכרו שבמודל תת-לינארי יש שתי סרטים: סרט קלט (לקריאה בלבד) וסרט עבודה (קריאה/כתיבה). במודל זה אנו נייצג קונפיגורציה באמצעות מיקום הראש הקורא בסרט הקלט, מיקום הראש הקורא בסרט העבודה, מצב נוכחי ותוכן סרט העבודה. לפיכך, מספר הקונפיגורציות במקרה כזה הוא $|Q_M| \cdot |\Gamma_M|^{s(n)} \cdot s(n) \cdot n = 2^{O(s(n))} \cdot n$. במקרה ש-$s(n) = \log n$ אנו עדיין מקבלים שקיים קבוע $d > 0$ כך שמספר הקונפיגורציות של M חסום ע"י $2^{d \cdot s(n)}$. יתר ההוכחה זהה.

- משפט סביץ' **לא גורר** ש- $\mathsf{L} \subseteq \mathsf{NL}$. משפט סביץ' רק מספר לנו ש-$\mathsf{NSPACE}(\log n) \subseteq \mathsf{SPACE}\left(\log^2 n\right)$.

25.2 שלמות ב-NL

בטרם נגדיר את מושג השלמות ב-NL, נתוודע לסוג חדש של רדוקציות: רדוקציות בשטח לוגריתמי.

הגדרה 25.4. משרן לוגריתמי (log space transducer) הוא מכונת טיורינג בעלת שלושה סרטים:

- סרט קלט לקריאה בלבד (read only).
- סרט עבודה, read/write, שמכיל לכל היותר $O(\log n)$ אותיות. (n - גודל הקלט)
- סרט פלט לכתיבה בלבד (write only). בפרט זה אומר שלא ניתן לקרוא אותיות שכבר כתבנו.

נאמר כי $f : \Sigma^* \to \Sigma^*$ היא **פונקציה ניתנת לחישוב בשטח לוגריתמי** אם"ם קיים משרן לוגריתמי M שבהינתן מילת קלט w כותב את $f(w)$ על סרט הפלט.
שפה A ניתנת **לרדוקציה בשטח לוגריתמי** לשפה B אם"ם קיימת פונקציה f שניתנת לחישוב בשטח לוגריתמי, כך שמתקיים:

$$w \in A \iff f(w) \in B$$

במקרה זה נסמן $A \leq_{\mathrm{L}} B$.

הבעיה האם $\mathsf{NL} \stackrel{?}{=} \mathsf{L}$ היא בעיה פתוחה במדעי המחשב. בדומה לתהליך שעברנו עם המחלקות P ו-NP, נגדיר כעת את מושג השלמות ב-NL:

הגדרה 25.5 (שלמות ב-NL)**.** שפה B תיקרא NL-**שלמה** אם"ם:

1. $B \in \mathsf{NL}$.
2. B היא NL-קשה: לכל שפה $A \in \mathsf{NL}$ מתקיים $A \leq_{\mathrm{L}} B$.

26 שבוע 15 - 24.06.19

26.1 שלמות ב-NL - המשך

בתרגול נוכיח את המשפט הבא:

משפט 26.1 (משפט הרדוקציה לרדוקציות בשטח לוגריתמי). יהיו $A, B \subseteq \Sigma^*$ שפות כך ש- $A \leq_{\text{L}} B$.

- אם $B \in \mathsf{L}$ אז $A \in \mathsf{L}$.
- אם $B \in \mathsf{NL}$ אז $A \in \mathsf{NL}$.

מסקנה ישירה מהמשפט היא:

מסקנה 26.2. אם B היא NL-קשה ו-$B \in \mathsf{L}$ אז $\mathsf{NL} = \mathsf{L}$.

הוכחה. נראה את הכיוון $\mathsf{NL} \subseteq \mathsf{L}$ (הכיוון השני ברור). תהי $A \in \mathsf{NL}$. הואיל ו-B היא NL-קשה מתקיים $A \leq_{\text{L}} B$. מאחר ש-$B \in \mathsf{L}$ נקבל ממשפט הרדוקציה לרדוקציות בשטח לוגריתמי כי $A \in \mathsf{L}$. □

הערה 26.3. משפט 26.1 מהווה מוטיבציה להגדרת המושג רדוקציה בשטח לוגריתמי (כפי שהוגדר בהגדרה 25.4). נשים לב שאם היינו משתמשים ברדוקציות פולינומיות במקום ברדוקציות בשטח לוגריתמי אז ההוכחה "הסטנדרטית" של משפטים מהסגנון הזה (לדוגמה, משפטים 14.6 ו-19.3) לא הייתה נכונה. יתר על כן, גם כאשר מושג ה-NL-קשות מוגדר באמצעות רדוקציות שפועלות בשטח לוגריתמי, ההוכחה "הסטנדרטית" לא תעבוד. נרחיב על כך. נניח כי B היא NL-קשה וגם $B \in \mathsf{L}$. תהי A שפה כלשהי ב-NL ותהי M_B מכונת טיורינג דטרמיניסטית המכריעה את B בשטח לוגריתמי (יש כזו כי $B \in \mathsf{L}$). בנוסף, $A \leq_{\text{L}} B$ ולכן קיימת פונקציית מיפוי f שניתנת לחישוב בשטח לוגריתמי. נבנה מכונה דטרמיניסטית M_A שפועלת בשטח לוגריתמי כך: בהינתן קלט w, M_A תחשב את $f(w)$ ותריץ את M_B עליה. תוכלו לזהות את הבעיה בטיעון? אמנם הפונקציה f ניתנת לחישוב בשטח לוגריתמי, אך אין שום מגבלה על אורך הפלט שלה. במילים אחרות, $|f(w)|$ לא בהכרח לוגריתמי ב-$|w|$, ולכן אין ל-M_A מספיק מקום כדי לכתוב את $f(w)$ על גבי סרט העבודה שלה. פותרים את הבעיה על ידי חישוב $f(w)$ "לפי דרישה" של M_B. הפרטים המלאים, כאמור, בתרגול.

26.2 השפה PATH היא NL-שלמה

משפט 26.4. PATH היא NL-שלמה.

הוכחה. בדוגמה 25.3 ראינו כי $\mathsf{PATH} \in \mathsf{NL}$. נוכיח כעת כי לכל $B \in \mathsf{NL}$ מתקיים $B \leq_{\text{L}} \mathsf{PATH}$. כיוון ש-$B \in \mathsf{NL}$ אנו יודעים שקיימת מ"ט אי-דטרמיניסטית M_B שמכריעה את B בשטח לוגריתמי. תהי $s(n)$ פונקציית סיבוכיות המקום של M_B. עלינו לבנות פונקציה ניתנת לחישוב f שבהינתן $w \in \Sigma^*$ מייצרת $\langle G, s, t\rangle$ בשטח לוגריתמי כך ש- $w \in B \iff$ יש ב-G מסלול מ-s ל-t. הרעיון הוא ש-G יהיה גרף הקונפיגורציות של M_B על w. בגרף קונפיגורציות, הקודקודים הם קונפיגורציות ויש צלע בין קונפיגורציה C ל-C' אם"ם הקונפיגורציות עוקבות. הקודקוד s יהיה הקונפיגורציה ההתחלתית; הקודקוד t יהיה הקונפיגורציה המקבלת. נניח בלי הגבלת הכלליות שקיימת קונפיגורציה מקבלת יחידה. אם הצלחנו לבנות את $\langle G, s, t\rangle$ כנ"ל בשטח לוגריתמי, הרדוקציה בוודאי נכונה: $w \in B$ אם"ם קיים חישוב מקבל של M_B על w אם"ם קיים מסלול בין s ל-t בגרף הקונפיגורציות G אם"ם $\langle G, s, t\rangle \in \mathsf{PATH}$.

כעת נראה שהרדוקציה ניתנת לבניה בשטח לוגריתמי. תחילה, נבין מהי קונפיגורציה בהקשר זה. קונפיגורציה של M_B על w ($|w| = n$) מכילה את (1) מיקום הראש של סרט הקלט (2) תוכן סרט העבודה (3) מיקום הראש של סרט העבודה (4) המצב הנוכחי לכן, נתאר קונפיגורציות על ידי מילה מהצורה $\gamma_1\gamma_2 \ldots (q, \gamma_i) \ldots \gamma_{s(n)}\$j$, כאשר $\gamma_i \in \Gamma_B$, $q \in Q_B$ ו-j הוא מספר בין 1 ל-n שמציין את מיקום הראש של סרט הקלט. נשים לב שניתן לייצג את j ע"י $\log n$ ביטים. זאת אומרת, קונפיגורציה היא מילה מעל $(\Gamma_B \cup (Q_B \times \Gamma_B))^{s(n)}\ \$\ (0 \cup 1)^{\log n}$. המכונה שמחשבת את f פועלת על w כך:

- עוברת על כל המילים ב-$(\Gamma_B \cup (Q_B \times \Gamma_B))^{s(n)}\ \$\ (0 \cup 1)^{\log n}$; אם מילה מתארת קונפיגורציה חוקית, תעתיק אותה לסרט הפלט. בסוף שלב זה כתובים כל **קודקודי** G על גבי סרט הפלט.
- המכונה עוברת על כל המילים ב-$(\Gamma_B \cup (Q_B \times \Gamma_B))^{s(n)}\ \$\ (0 \cup 1)^{\log n}\ \#\ (\Gamma_B \cup (Q_B \times \Gamma_B))^{s(n)}\ \$\ (0 \cup 1)^{\log n}$; אם מילה $C\#C'$ מקודדת שתי קונפיגורציות עוקבות אז היא תועתק לסרט הפלט. בסוף שלב זה כתובות **הצלעות** של G על גבי סרט הפלט.
- המכונה כותבת את הקידוד של s ו-t על גבי סרט הפלט. זכרו כי s היא הקונפיגורציה ההתחלתית של M_B על w ולכן מהצורה $(q_0, w_1)\, w_2 \ldots w_n \_\_ \ldots \_\$000 \ldots 0$ כאשר $w = w_1 \ldots w_n$. הזכרנו קודם שניתן להניח בלי הגבלת הכלליות שקיימת קונפיגורציה מקבלת יחידה, לכן t מוגדר היטב. (אם בבוא המכונה לקבל את w היא מוחקת את תוכן סרט העבודה ומזיזה את הראשים הקוראים שמאלה לתחילת הסרט, אז t הוא מהצורה $(q_{acc}, \_)\ \_ \ldots \_\$000 \ldots 0$.)

מעבר על כל המילים שמתארות קונפיגורציות יכול להתבצע בשטח לוגריתמי (למשל, עוברים על המילים בסדר לקסיקוגרפי). בנוסף, מכיוון ש-$s(n) = O(\log n)$ אנו יודעים שהייצוג של קונפיגורציה הוא $O(\log n)$. לפיכך, בכל שלב המכונה משתמשת במספר תאים שהוא $O(\log n)$. מכאן שהרדוקציה פועלת בשטח לוגריתמי, כנדרש. □

משפט 26.5. $\mathsf{NL} \subseteq \mathsf{P}$.

הוכחה. בהינתן בעיה $B \in \mathsf{NL}$ אנו יודעים כי $B \leq_{\mathsf{L}} \mathsf{PATH}$ שהרי PATH היא NL-שלמה. בנוסף, הרדוקציה שהראינו בהוכחת משפט 26.4 עובדת גם בזמן פולינומיאלי, ולכן $B \leq_p \mathsf{PATH}$. אנו יודעים ש-$\mathsf{PATH} \in \mathsf{P}$ (PATH ניתנת לפתרון בזמן פולינומיאלי ע״י אלגוריתמי חיפוש בגרף כגון BFS ו-DFS), ולכן $B \in \mathsf{P}$. לפיכך, $\mathsf{NL} \subseteq \mathsf{P}$, כנדרש. □

דוגמה 26.6 (השפה BAR). נתבונן בשפה BAR (קיצור של bounded above reachability) שמוגדרת באופן הבא:

$$\mathsf{BAR} := \left\{ \langle G, s, t, b \rangle \mid {}^{G \text{ is weighted directed graph, with positive natural weights}}_{\text{s.t. there exists a path from } s \text{ to } t \text{ that weights at most } b} \right\}$$

(נזכיר שגרף ממושקל עם משקלים חיוביים הוא שלשייה $G = (V, E, w)$ כאשר $w : E \to \mathbb{N}$). נוכיח ש-BAR היא NL-שלמה:

- $\mathsf{NL} \in \mathsf{BAR}$: מכונת טיורינג אי-דטרמיניסטית שמכריעה את BAR בשטח לוגריתמי תנחש מסלול מ-s ל-t ובכל צעד תחזיק קודקוד נוכחי v ומחיר מצטבר p. ביתר פירוט:
  1. בשלב האתחול נבצע $v := s$ ו-$p := 0$.
  2. כל עוד $p \leq b$ נבצע:
     (א) אם $v = t \Leftarrow$ עצור וקבל.
     (ב) אחרת, המכונה תנחש קודקוד u. אם אין קשת מ-v ל-u המכונה תעצור ותדחה.
     (ג) המכונה תעדכן $v \leftarrow u$ ו-$p \leftarrow p + w(v, u)$.
  3. דחה.

  נבחין כי המכונה עוצרת בכל החישובים שלה מאחר שהמשקלים על הצלעות חיוביים (לכן בכל שלב הערך של p גדל ממש).

- BAR היא NL-קשה: נראה רדוקציה $\mathsf{PATH} \leq_{\mathsf{L}} \mathsf{BAR}$. בהינתן $\langle G, s, t \rangle$ עלינו להחזיר $\langle G', s', t', b \rangle$ כאשר $\langle G, s, t \rangle \in$ PATH אם ורק אם $\langle G', s', t', b \rangle \in \mathsf{BAR}$. הרדוקציה תפעל כך:

$$\begin{aligned} G' &:= \langle V(G), E(G), w \rangle \text{ where } w(v,u) = 1 \text{ for all } (v,u) \in E(G) \\ s' &:= s \\ t' &:= t \\ b &:= |V(G)| \end{aligned}$$

  הרדוקציה בוודאי ניתנת למימוש בשטח לוגריתמי. בנוסף, קיים מסלול ב-G בין s ל-t אס״ם קיים מסלול פשוט ב-G מ-s ל-t אס״ם קיים מסלול ב-G' בין s' ל-t' ששוקל לכל היותר b.

דוגמה 26.7 (וריאציות של השפה BAR). נתאר בקצרה מספר שפות דומות לשפה BAR:

1. השפה BBR מוגדרת כך: $\mathsf{BBR} := \left\{ \langle G, s, t, b \rangle \mid {}^{G \text{ is weighted directed graph, with positive natural weights}}_{\text{s.t. there exists a path from } s \text{ to } t \text{ that weights } \textbf{at least } b} \right\}$. ניתן להוכיח כי $\mathsf{BBR} \in \mathsf{NL}$ בדומה להוכחה ש-$\mathsf{BAR} \in \mathsf{NL}$. בנוסף, BBR היא NL-קשה כי ניתן להראות רדוקציה $\mathsf{PATH} \leq_{\mathsf{L}} \mathsf{BBR}$. הרדוקציה תעבוד כך: בהינתן $\langle G, s, t \rangle$ הרדוקציה תחזיר את אותו הגרף עם משקלים 1 על כל הצלעות וחסם תחתון $b = 0$.

2. השפה UBBR מוגדרת כך:

$$\mathsf{UBBR} := \{ \langle G, s, t \rangle \mid \forall b \in \mathbb{N}, \text{ there exists a path in } G \text{ from } s \text{ to } t \text{ that weights at least } b \}$$

איפיון פשוט לשפה UBBR: $\langle G, s, t \rangle \in \mathsf{UBBR}$ אם ורק אם קיים קודקוד v עם התכונות הבאות: (א) v ישיג מעצמו, (ב) קיים מסלול בין s ל-v, (ג) קיים מסלול בין v ל-t. לאור איפיון זה, $\mathsf{UBBR} \in \mathsf{NL}$, שהרי מכונת טיורינג אי-דטרמיניסטית תנחש קודקוד v כנ״ל. בנוסף, נטען כי UBBR היא NL-קשה. לשם כך, נראה רדוקציה $\mathsf{PATH} \leq_{\mathsf{L}} \mathsf{UBBR}$. בהינתן $\langle G, s, t \rangle$ נייצר $\langle G', s', t' \rangle$ כאשר G' מתקבל מ-G ע״י הוספת חוג עצמי במשקל 1 ב-s (וכל המשקלים האחרים ב-G' יוגדרו גם ל-1).

הגדרה 26.8. המחלקה co-NL מוגדרת כך: $\text{co-}\mathsf{NL} := \left\{ \overline{L} \mid L \in \mathsf{NL} \right\}$.

המשפט הבא הוכח ע״י ניל אימרמן בשנת 1987 והובא ללא ההוכחה:

משפט 26.9 (משפט אימרמן). $\mathsf{NL} = \text{co-}\mathsf{NL}$.