מודלים חישוביים, חישוביות וסיבוכיות

ססה גולדשטיין, sashag@cs

20 ביוני 2011

תקציר

הסיכום להלן מהווה תקציר של חומר הקורס ואיני נוטל עליו כל אחריות. אתם יכולים להיעזר גם בהקלטות השיעורים וכמובן בספר הלימוד. החומר מבוסס על הרצאותיו של ד"ר גיא קינדלר ושיעורי התרגול של מר רועי פוקס.

ניתן למצוא את הגרסה המעודכנת ביותר של סיכום זה ב־ Notes־Heaven. אשמח לקבל הערות ותיקונים לסיכום, גם אחרי מועד המבחן, לכתובת המייל שכתובה למעלה.

תוכן עניינים

להלן מספר דוגמאות לבעיות חישוביות. במהלך הקורס ננסה להפריד בין הבעיות האלה:

1. בהנתן גרף G, רוצים לבדוק האם יש בו מסלול אוילר.
2. נתונים המספרים $a_1, \ldots, a_n$ ורוצים לבדוק האם ניתן להציג את t כסכום של ת"ק של $a_1, \ldots, a_n, t$.
3. נתונות שתי סדרות מחרוזות בינאריות $x_1, \ldots, x_n, y_1, \ldots, y_n$ ורוצים לבדוק האם יש סדרת אינדקסים $i_1, \ldots, i_k$ כך ש־ $x_{i_1}, \ldots, x_{i_k} = y_{i_1}, \ldots, y_{i_k}$.
4. בהנתן טענה מתמטית, רוצים לבדוק האם יש לה הוכחה.
5. בהנתן תוכנית מחשב וקלט, רוצים לבדוק האם התוכנית נכנסת ללולאה אינסופית בריצתה על הקלט.

במהלך הקורס נראה שהבעיה הראשונה "קלה" מבחינה חישובית, השנייה "קשה" אך עדיין ניתנת לפתרון, ושלוש הבעיות האחרונות אינן ניתנות לפתרון באמצעות מחשב.

חלק I
מודלים חישוביים

1 מבוא לשפות פורמליות

בפרק זה נראה מספר מושגי יסוד.

1.1 קבוצות וגדלים של קבוצות

הגדרה 1.1 $|A| \leq |B|$ אם קיימת $f : A \to B$ חח"ע.
$|A| = |B|$ אם $|A| \leq |B|$ וגם $|B| \leq |A|$.
$|A| < |B|$ אם $|A| \leq |B|$ וגם $|A| \neq |B|$.

משפט 1.2 (קנטור־ברנשטיין) $|A| = |B|$ אסם קיימת $f : A \to B$ חח"ע ועל.

הגדרה 1.3 $|\mathbb{N}| = \aleph_0$ ונאמר שאם $|A| = \aleph_0$ אז A בת־מניה.

הערה 1.4 מתקיים איפוא ש־ $|\mathbb{Z}| = |\mathbb{Q}| = \aleph_0$.

הגדרה 1.5 $|[0,1]| = \aleph$

טענה 1.6 $\aleph_0 < \aleph$

הוכחה: קל לראות ש־ $\aleph \leq \aleph_0$ ע"י ההתאמה $f : n \mapsto \frac{1}{n}$.
נשתמש בטיעון האלכסון של קנטור כדי להראות שלא קיימת פונקציה על $f : \mathbb{N} \to [0,1]$. נראה זאת אפילו לגבי $[0,1)$. נכתוב בשורה את ערכי f:

$$\begin{aligned} f(1) &= 0.a_{11}a_{12}a_{13}\ldots \\ f(2) &= 0.a_{21}a_{22}a_{23}\ldots \\ f(3) &= 0.a_{31}a_{32}a_{33}\ldots \\ \vdots &\quad \vdots \quad \vdots \end{aligned}$$

וכעת נגדיר מספר חדש $x = 0.b_1b_2b_3\ldots$ עם הדרישה $b_i \neq a_{ii}$. ברור שאין ל־ x מקור ביחס ל־ f ולכן היא אינה על. ■

1.2 שפות פורמליות

הגדרה 1.7 א"ב Σ זו קבוצה סופית לא ריקה, שלאיבריה נקרא אותיות או תווים.
מילה מעל Σ היא המילה הריקה (ללא אותיות) המסומנת ε או איבר ב־ Σ^n עבור $n \in \mathbb{N}$ כלשהו.
אם u, w מילים מעל Σ אז המילה uw (לפעמים נסמן $u \cdot w$) היא השרשור שלהן.
Σ^* זהו אוסף כל המילים מעל Σ כולל המילה הריקה. (כלומר, $\Sigma^* = \bigcup_{n \in \mathbb{N} \cup \{0\}} \Sigma^n$)
שפה מעל Σ היא תת־קבוצה של Σ^*.

דוגמאות לשפות:

1. השפה הריקה (מעל א"ב כלשהו).
2. שפת כל המילים (מעל א"ב כלשהו).
3. שפת המילים מעל $\{a, b\}$ באורך 23 לכל היותר.
4. שפת המילים מעל $\{a, b\}$ שבהן שלוש אותיות זהות רצופות.
5. שפת המספרים העשרוניים החיוביים השלמים המתחלקים ב־3.
6. שפת המספרים הראשוניים.
7. שפת הטענות המתמטיות שיש להן הוכחה במערכת ההיסק של פרגה מאקסיומות ZFC.

הגדרה 1.8 אם $L_1, L_2 \subseteq \Sigma^*$ נגדיר את הפעולות הבאות:

$$
\begin{aligned}
L_1 \cdot L_2 &= \{uw \ : w \in L_1,\ x \in L_2\} \\
L_1 \cap L_2 & \\
L_1 \cup L_2 & \\
\overline{L_1} &= \Sigma^* \setminus L_1 \\
L_1^* &= \bigcup_{n \in \mathbb{N} \cup \{0\}} L_1^n
\end{aligned}
$$

דוגמאות לפעולות על שפות:
יהיו L_1 שפת המילים מעל $\{a, b\}$ המתחילות ב־ a ו־ L_2 שפת המילים מעל אותו א"ב המסתיימות ב־ b. אז $L_1 \cap L_2$ אלה מילים המתחילות ב־ a ומסתיימות ב־ b. במקרה ספציפי זה, גם $L_1 \cdot L_2$ זה בדיוק אותו הדבר.

הערה 1.9 יהי Σ א"ב לא ריק. אזי $|\Sigma^*| = \aleph_0$ ו־ $|2^{\Sigma^*}| = \aleph$.

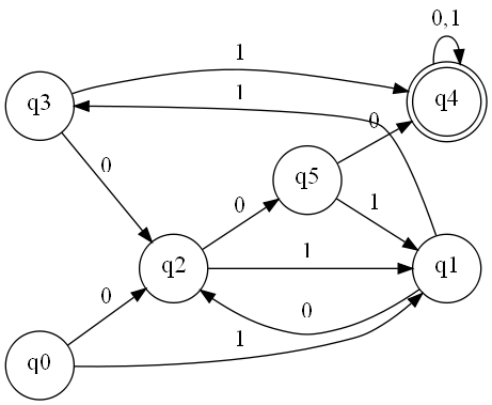
2 אוטומטים

האוטומטים שנציג בפרק זה הם מודלים חישוביים פשוטים יחסית שלא מתקרבים ליכולתו של מחשב.

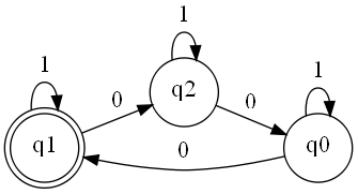
2.1 אוטומטים סופיים דטרמיניסטיים

הגדרה 2.1 אוטומט סופי דטרמיניסטי (DFA) הוא חמישייה סדורה $A = (Q, \Sigma, \delta, q_0, F)$ כאשר Q קבוצת מצבים סופית, Σ הוא א"ב, $q_0 \in Q$ המצב ההתחלתי, $F \subseteq Q$ קבוצת המצבים המקבלים, ו־ $\delta : Q \times \Sigma \to Q$ פונקציית המעברים.

דוגמאות לאוטומטים:
אוטומט שמקבל את המילים שבהן ביט שחוזר על עצמו שלוש פעמים ברציפות:



אוטומט שמקבל את המילים שבהן מספר האפסים מתחלק ב־3 עם שארית 1:



הגדרה פורמלית של האוטומט הזה: $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $q_0 = q_0$, $F = \{q_1\}$ ונותר להגדיר את פונקציית המעברים.
נעשה זאת באמצעות טבלה:

$\delta(q, \sigma)$	$\sigma = 0$	$\sigma = 1$
$q = q_0$	q_1	q_0
$q = q_1$	q_2	q_1
$q = q_2$	q_0	q_2

בדרך כלל נסתפק בציור של האוטומט.

הגדרה 2.2 בהנתן A עם פונקציית מעברים δ, נגדיר $\delta^* : Q \times \Sigma^* \to Q$ באינדוקציה על האורך של $w \in \Sigma^*$:
אם $w = \varepsilon$ אז $\delta^*(q, w) := q$ לכל $q \in Q$;
אם הגדרנו את δ^* עבור מילים באורך n ו־ w היא מילה באורך $n + 1$, אז נכתוב את $w = ua$ כאשר $a \in \Sigma$ ונגדיר $\delta^*(q, w) = \delta^*(q, ua) := \delta(\delta^*(q, u), a)$.

הערה 2.3 נאמר שבריצתו על w האוטומט מגיע למצב q אם $\delta^*(q_0, w) = q$. האוטומט <u>מקבל</u> את המילה אם $\delta^*(q_0, w) \in F$.

הגדרה 2.4 יהי A אוטומט סופי דטרמיניסטי מעל Σ. השפה L המתקבלת ע"י A היא

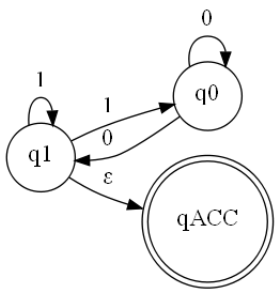
$$L := \{w \in \Sigma^* : \ \delta^*(q_0, w) \in F\}$$

שפה המתקבלת ע"י DFA כלשהו נקראת <u>רגולרית</u>.
אוסף השפות הרגולריות יסומן REG.

2.2 אוטומטים סופיים לא דטרמיניסטיים

הגדרה 2.5 אוטומט סופי לא דטרמיניסטי (NFA) הוא חמישייה סדורה $A = (Q, \Sigma, \delta, Q_0, F)$ כאשר Q קבוצת מצבים סופית, Σ א"ב, $Q_0 \subseteq Q$ קבוצת המצבים ההתחלתיים, $F \subseteq Q$ קבוצת המצבים המקבלים, ו־ $\delta : Q \times \Sigma \to 2^Q$ פונקציית המעברים.

<u>דוגמא לאוטומט סופי לא דטרמיניסטי</u>:



הגדרה 2.6 בהנתן A עם פונקציית מעברים δ, נגדיר $\delta^* : 2^Q \times \Sigma^* \to 2^Q$ באינדוקציה על האורך של $w \in \Sigma^*$:
אם $w = \varepsilon$ אז $\delta^*(S, w) := S$ לכל $S \subseteq Q$;
אם הגדרנו את δ^* עבור מילים באורך n ו־ w היא מילה באורך $n+1$, אז נכתוב את $w = ua$ כאשר $a \in \Sigma$ ונגדיר

$$\delta^*(S, w) = \delta^*(S, ua) := \bigcup_{q \in \delta^*(S,u)} \delta(q, a)$$

הערה 2.7 נאמר שהאוטומט מקבל את המילה $w \in \Sigma^*$ אם $\delta^*(Q_0, w) \cap F \neq \emptyset$.

הגדרה 2.8 ריצה חוקית של A על המילה $w = (w_1, \ldots, w_n)$ היא סדרת מצבים $q_1, \ldots, q_n \in Q$ כך ש־ $q_0 \in Q_0$ וגם $\forall 1 \leq i \leq n,\ q_i \in \delta(q_{i-1}, w_i)$.

טענה 2.9 יהי A NFA מעל Σ. $w \in L(A)$ אמ"מ קיימת ריצה חוקית של A על w שהמצב האחרון בה שייך ל־ F.

הוכחה: טריוויאלית, אינדוקציה על אורך המילה. ■

הערה 2.10 בהגדרה שלנו של NFA (כשמציירים אותו על דף) אנו מרשים מעברי ε. ניתן להשתכנע (ר' תרגול 2) שאין צורך אמיתי במעברים אלה. למשל, אם $\delta(q, a) = \{q'\}$ ויש מעבר ε בין q' ל־ q'', אנו יכולים להרחיב את δ כך ש־ $\delta(q, a) = \{q', q''\}$. כלומר, השוני המרכזי של NFA מ־ DFA אינו מעברי ε, אלא היכולת "להתפצל".

משפט 2.11 יהי $A = (Q_A, \Sigma, \delta_A, Q_0, F_A)$ אוטומט סופי דטרמיניסטי. אזי קיים אוטומט סופי דטרמיניסטי $B = (Q_B, \Sigma, \delta_B, q_0, F_B)$ המקיים $L(B) = L(A)$. יתר על כן, קיים B כזה שעבורו $|Q_B| \leq 2^{|Q_A|}$, והחסם הדוק.

הוכחה: נגדיר את B כדלהלן. $Q_B = 2^{Q_A}$, $q_0 = Q_0$. את פונקציית המעברים נגדיר כך:

$$\begin{aligned} \delta_B : Q_B \times \Sigma &\to Q_B \\ \delta_B(S, a) &= \delta_A^*(S, a) \end{aligned}$$

ולבסוף $F_B = \{S \in 2^{Q_A} :\ S \cap F_A \neq \emptyset\}$. כעת יש להראות באינדוקציה ש־ $\delta_B^*(S, w) = \delta_A^*(S, w)$ ומכאן מתקבל הדרוש:

$$w \in L(A) \iff \delta_A^*(Q_0, w) \cap F_A \neq \emptyset \iff \delta_B^*(q_0, w) \in F_B \iff w \in L(B)$$

■

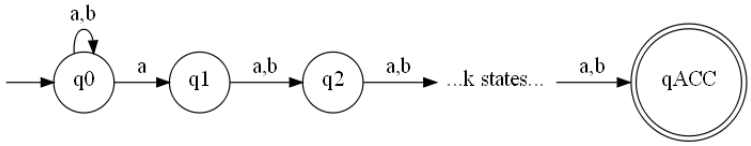
<u>דוגמה לדטרמיניזציה</u>:
ר' תרגיל 2 שאלה 3ב'.

טענה 2.12 החסם במשפט הדוק, כלומר יש משפחה אינסופית של שפות שה־ DFA המינימלי שמזהה אותן הוא בעל מספר אקספוננציאלי של מצבים לעומת ה־ NFA השקול.

הוכחה: המשפחה היא

$$L_k = \{wau: \quad w,u \in \{a,b\}^*,\ |u| = k-1\}$$

ה־ NFA שמזהה את L_k נראה כך:



בסה"כ באוטומט הזה יש $k+1$ מצבים. לעומת זאת, ב־ DFA הקטן ביותר שמזהה את L_k יש לפחות 2^k מצבים. אינטואיטיבית, האוטומט צריך לזכור בכל מצב האם היה a עד אותו שלב או לא (לפחות בחלון של k אותיות אחורה צריך לזכור את כל האותיות שנקראו). מספר האפשרויות האלה הוא 2^k.

פורמלית: נתבונן בכל המילים Σ^k שמספרן 2^k במקרה שלנו, ונניח בשלילה שיש DFA A_k שבו $|Q| < 2^k$. אז כעת יש $y \neq x \in \Sigma^k$ כך ש־ $\delta^*(q_0,x) = \delta^*(q_0,y)$ משיקולי שובך היונים.

מכך ש־ $x \neq y$ קיים i כך ש־ $x_i \neq y_i$. בה"כ $x_i = a$ ו־ $y_i = b$. נוסיף לשתי המילים את הסיפא $z = b^{k-1}$. מתקיים ש־ $\delta^*(q_0,xz) = \delta^*(q_0,yz)$ אבל ברור ש־ $xz \in L$ ו־ $yz \notin L$. זו הסתירה. ■

הערה 2.13 לכאורה, בדרך כלל אי־דטרמיניסטיות עוזרת לנו. אבל יש מקרים (גם פשוטים) שבהם לא ניתן לצמצם אוטומט דטרמיניסטי ע"י מעבר לאוטומט לא דטרמיניסטי שקול. למשל, האוטומט הדטרמיניסטי שמקבל את $L = \{a^k\}$ הוא בעל $k+1$ מצבים, וניתן להוכיח (באמצעות ניפוח) שגם באוטומט לא דטרמיניסטי שמקבל את L חייבים להיות לפחות $k+1$ מצבים.

2.3 פעולות על שפות רגולריות

בהנתן L, M שפות רגולריות מעל א"ב Σ, נרצה לבדוק האם $L \cap M,\ L \cup M,\ L \cdot M,\ \overline{L},\ L^*$ הן רגולריות.

טענה 2.14 $L \in REG \implies \overline{L} \in REG$

הוכחה: יש אוטומט $A = (Q,\Sigma,\delta,q_0,F)$ המקבל את L. האוטומט $A' = (Q,\Sigma,\delta,q_0,Q \setminus F)$ זהו האוטומט המבוקש המקבל את $\overline{L}$ ולכן היא רגולרית. (כמובן, הוכחה זו לא שלמה.) ■

טענה 2.15 $L, M \in REG \implies L \cap M \in REG$

הוכחה: יהיו $A = (Q,\Sigma,\delta_A,q_0,F_A)$ ו־ $B = (R,\Sigma,\delta_B,r_0,F_B)$ אוטומטים המזהים את L, M בהתאמה. אזי האוטומט $C = (Q \times R,\Sigma,\delta_C,(q_0,r_0),F_A \times F_B)$ (המכונה גם "אוטומט המכפלה") עם

$$\delta_C((q,r),a) = (\delta_A(q,a),\delta_B(r,a))$$

מקבל בדיוק את $L \cap M$. תחילה נראה באינדוקציה ש־

$$\delta_C^*((q_0,r_0),w) = (\delta_A^*(q_0,w),\delta_B^*(r_0,w))$$

(זה טריוויאלי). כעת, אם $w \in L \cap M$ אז $\delta_A^*(q_0,w) \in F_A$ ו־ $\delta_B^*(r_0,w) \in F_B$ ולכן $\delta_C^*((q_0,r_0),w) \in F_A \times F_B$ ולהיפך. ■

טענה 2.16 $L, M \in REG \implies L \cup M \in REG$

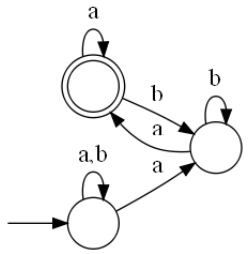
הוכחה: כמו הטענה הקודמת, אלא שקבוצת המצבים המקבלים של C תהיה $(F_A \times R) \cup (Q \times F_B)$. ■

טענה 2.17 $L, M \in REG \implies L \cdot M \in REG$

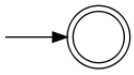
הוכחה: יהיו $A = (Q, \Sigma, \delta_A, q_0, F_A)$ ו־ $B = (R, \Sigma, \delta_B, r_0, F_B)$ אוטומטים המזהים את L, M בהתאמה. נבנה $C = \text{NFA}$ $(Q \cup R, \Sigma, \delta', q_0, F_B)$ כאשר δ' מתנהגת כמו δ_A ו־ δ_B, אבל בנוסף מאפשרת מעבר ε מכל $q \in F_A$ ל־ r_0.
ברור מהבנייה של האוטומט שאם מילה היא ב־ $L \cdot M$ אז ישנה ריצה של האוטומט שמקבלת אותה, ולהיפך - אם האוטומט מקבל את המילה, אפשר לשבור אותה לשני חלקים בדיוק סביב מעבר ה־ ε שבוצע עליה (שהרי אין שום חיבור אחר בין Q ל־ R).■

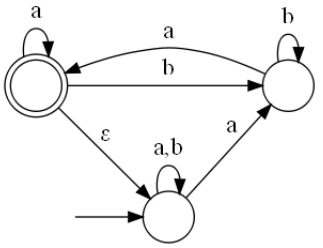
טענה 2.18 $L \in REG \implies L^* \in REG$

הוכחה: ניקח NFA שמקבל את L ונבנה ממנו NFA שנראה אותו הדבר, אבל δ מחברת במעבר־ε כל מצב מקבל שלו לכל מצב התחלתי שלו. כדי לקבל גם את ε נוסיף מצב מקבל והתחלתי ללא כל מעברים ממנו.
לדוגמא, עבור האוטומט



האוטומט החדש הוא





■

2.4 למת הניפוח לשפות רגולריות

נרצה להראות שקיימות גם שפות לא רגולריות. הדבר ברור משיקולי ספירה (קבוצת האוטומטים המגדירים שפות רגולריות היא בת־מניה, בעוד שקבוצת השפות אינה בת־מניה), אבל אנו מחפשים דרך לטעון על שפה ספציפית שהיא לא רגולרית.

טענה 2.19 יהי $\Sigma = \{0, 1\}$ ו־ $L = \{0^n 1^n : n \in \mathbb{N}\}$. אזי $L \notin REG$.

הוכחה: אינטואיטיבית זה ברור - האוטומט צריך "לזכור" כמה אפסים וכמה אחדים היו עד עכשיו, וזה דורש אינסוף מצבים. פורמלית:
נניח בשלילה ש־ $L \in REG$ והאוטומט $A = (Q, \Sigma, \delta, q_0, F)$ מקבל אותה, ויהי $k = |Q|$. נתבונן על ריצת האוטומט על המילה $0^{k+1}1^{k+1} \in L$. האוטומט עובר בין $k+1$ מצבים במהלך הטיפול באפסים, ואז עוד $k+1$ מצבים במהלך הטיפול באחדים.
ב־ $k+1$ המצבים הראשונים יש מצב אחד שחוזרים עליו לפחות פעמיים (שובך היונים), ונסמן אותם $q_i = q_j$ כאשר $i < j$. כלומר, אחרי 0^i הגענו למצב q_i ולאחר עוד 0^{j-i} שוב הגענו למצב q_i. כלומר, הריצה שלנו על המילה $0^i 0^{j-i} 0^{k-j} 1^k$ צריכה להיות זהה לריצה על $0^i 0^{j-i} 0^{j-i} 0^{k-j} 1^k$, אז אנחנו אמורים לקבל גם אותה - אבל המילה הזאת אינה בשפה בכלל!
זוהי סתירה, ולכן נסיק שהשפה אינה רגולרית כנדרש. ■

משפט 2.20 (למת הניפוח, Pumping Lemma) אם L רגולרית אז

$$\exists p > 0 \quad \forall w \in L, |w| > p \quad \exists x, y, z \in \Sigma^* \quad w = xyz,\ |y| > 0,\ |xy| \leq p \qquad \forall i \in \mathbb{N} \cup \{0\} \quad xy^i z \in L$$

הוכחה: ההוכחה דומה למה שעשינו קודם, כאשר p ייבחר להיות $|Q| + 1$ עבור האוטומט. ■

<u>דוגמא לשימוש בלמת הניפוח</u>:

מעל $\Sigma = \{a, b\}$, נראה ש־ $L = \{uu : u \in \Sigma^*\}$ אינה רגולרית. נניח בשלילה שהיא רגולרית, ואז קיים $p > 0$ מהלמה. המילה $a^{p+1}ba^{p+1}b \in L$ מקיימת את תנאי הלמה, ולכן יש לה פירוק xyz כך ש־ xy מורכב מ־ a־ים בלבד, וגם $|y| > 0$.
נסמן $x = a^m$, $y = a^r$, $z = a^{p+1-r-m}ba^{p+1}b$. כעת אנחנו יכולים לייצר מילים שאינן בשפה, למשל המילה $xyyz$ לפי הלמה צריכה להיות בשפה, אולם $xyyz = a^{p+r+1}ba^{p+1}b$ וכמובן אינה בשפה. זוהי סתירה ולכן L אינה רגולרית.

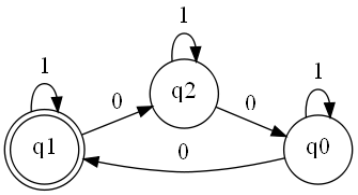
2.5 משפט Myhill־Nerode

למת הניפוח מהווה תנאי הכרחי לכך ששפה תהיה רגולרית, אך היא אינה תנאי מספיק. למשל, השפה : $L = \{w \in \{a, b\}^* : \#_a(w) \neq \#_b(w)\}$ אינה רגולרית, אך היא מקיימת את למת הניפוח. אנו רוצים תנאי הכרחי ומספיק לרגולריות.

הגדרה 2.21 תהי $L \subseteq \Sigma^*$. נאמר ש־ $u, w \in \Sigma^*$ שקולות ביחס ל־ L אם לכל $v \in \Sigma^*$ מתקיים ש־ $uv, wv \in L$ או $uv, wv \notin L$.
(בפרט עבור $v = \varepsilon$ ברור ש־ $v, w \in L$ או $v, w \notin L$.)
אם יש $v \in \Sigma^*$ שאינה מקיימת את התנאי הנ"ל, נאמר שהיא מפרידה בין u, w.
אם u, w שקולות ביחס ל־ L נסמן $u \sim_L w$.

הערה 2.22 היחס $\sim_L$ הוא יחס שקילות על מילים ב־ Σ^*.

דוגמאות ליחס השקילות:
תהי $L_1 = \{w : \#_a(w) \equiv 1 (\mathrm{mod} 3)\}$. מהן מחלקות השקילות ביחס ל־ L_1? יש שלוש מחלקות שקילות - $[\varepsilon] = [a],\ [aa],\ [aaa]$. ראינו כבר DFA מתאים, ולמעשה המצבים באוטומט "זוכרים" את מחלקת השקילות של המילה.



תהי $L_2 = \{w : \#_a(w) > \#_b(w)\}$. כאן כל $[a^k]$ עם $k \geq 0$ מגדירה מחלקת שקילות נפרדת. למשל $a \not\sim_L aa$ שכן המילה b מפרידה ביניהן. גם כל $[b^k]$ עם $k \geq 0$ מהווה מחלקת שקילות. למעשה, יש מחלקת שקילות לכל מספר $z \in \mathbb{Z}$ המכילה את המילים w כך ש־ $\#_a(w) - \#_b(w) = z$.

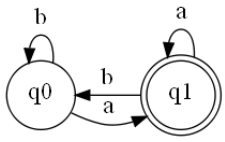
טענה 2.23 אם $L \subseteq \Sigma^*$ ו־ DFA A המקבל את L, וכן המילים $u, w \in \Sigma^*$ מקיימות $\delta^*(q_0, u) = \delta^*(q_0, w)$ אז $u \sim_L w$.

הוכחה: לכל $v \in \Sigma^*$ נקבל $\delta^*(q_0, uv) = \delta^*(q_0, wv)$, ולכן $uv \in L$ אסם $wv \in L$, כלומר $u \sim_L w$. ■

משפט 2.24 (Myhill־Nerode) $L \in REG$ אסם מספר מחלקות השקילות של היחס $\sim_L$ הוא סופי.

הוכחה: ($\Leftarrow$) תהי $L \in REG$, A DFA המזהה אותה, ו־ Q קבוצת המצבים שלו. נניח בה"כ שלכל $q \in Q$ יש מילה שנסמנה w_q כך ש־ $\delta^*(q_0, w_q) = q$. זה בה"כ כי מצבים שאי אפשר להגיע אליהם אפשר לסלק מהאוטומט ללא שינוי השפה שלו.
נסתכל על קבוצת מחלקות השקילות $\{[w_q] : q \in Q\}$. זו קבוצה סופית והיא מכילה את כל מחלקות השקילות של $\sim_L$, שכן לכל $w \in \Sigma^*$ אם $\delta^*(q_0, w) = q$ אז $w \in [w_q]$ לפי הטענה הקודמת. לכן מספר מחלקות השקילות הוא סופי.
($\Rightarrow$) תהי $L \subseteq \Sigma^*$ שמספר מחלקות השקילות שלה לפי $\sim_L$ הוא n. אז נראה ש־ L רגולרית, נציג DFA בעל n מצבים בדיוק המקבל את L, ונראה שאין DFA כזה בעל פחות מ־ n מצבים.
אם יש DFA עם פחות מ־ n מצבים שמקבל את L, אז יש פחות מ־ n מחלקות שקילות לפי הטענה הקודמת, וזו סתירה. כעת נבנה אוטומט עם המצבים $q_0, q_1, \ldots, q_{n-1}$ המתאימים למחלקות השקילות של $\sim_L$, כאשר $q_0 = [\varepsilon]$. נגדיר את $\delta(q, a) = \delta([w], a) = [wa]$. נשים לב שההגדרה לא תלויה בבחירת w כך ש־ $[w] = q$, שהרי אם גם $[w'] = q$ אז $w'a \sim_L wa$. אוסף המצבים המקבלים יהיה $\{\delta^*(q_0, w) : w \in L\}$; אפשר להסתכל על זה כמחלקות השקילות המכילות מילים בשפה (שהרי אם מחלקת שקילות מכילה מילה אחת בשפה, כל המילים בה הן בשפה).
נותר לשים לב ש־ $\delta^*(q_0, w) = [w]$ באינדוקציה קלה על אורך המילה.
כעת $w \in L$ אסם $[w]$ מכילה מילים ב־ L אסם $[w] \in F$ אסם $\delta^*(q_0, w) \in F$, כלומר אסם כשמריצים את A על w מגיעים למצב מקבל. לכן $L(A) = L$ כנדרש. ■

שימושים במשפט Myhill־Nerode:
1. נתבונן בשפה $L = \{a^k : k \notin \{2^0, 2^1, 2^2, \ldots\}\}$ מעל $\Sigma = \{a\}$. לבדיקת רגולריות נספור כמה מחלקות שקילות יש ליחס $\sim_L$. נתבונן במילים $x = a^{2^n}$ ו־ $y = a^{2^t}$ עבור $n < t$. יש להן סיפא מפרידה, למשל $z = a^{2^n}$ ואז $xz \in L$ אבל $yz \notin L$. לכן $[x] \neq [y]$. זה נכון לכל $n < t$ ולכן יש אינסוף מחלקות שקילות, כלומר L אינה רגולרית.
2. נתבונן בשפה $L = \{w \in \{a, b\}^* : w \text{ ends with } a\}$. נסתכל על מילים x, y ו־ z שעשויה להפריד ביניהן. אם z מסתיימת ב־ a אז $xz, yz \in L$ ואחרת $xz, yz \notin L$. כמו כן, יש שתי מחלקות שקילות שמופרדות ע"י ε - המילים בשפה והמילים שאינן בשפה. כל שאר ה־ z־ים לא יכולים להפריד בין מילים, ולכן L רגולרית, וכמובן האוטומט המזהה אותה הוא:



באוטומט זה המצב המקבל מתאים למחלקת השקילות L והמצב ההתחלתי מתאים למחלקת השקילות $\overline{L}$.

3. מינימיזציה של DFA: נתאר אלגוריתם המקבל DFA A ומייצר ממנו DFA שקול עם מספר מינימלי של מצבים. הפלט צריך להיות מחלקות השקילות ביחס ל־ $L(A)$, שמשרות את האוטומט שראינו בהוכחת משפט Myhill־Nerode.

נגדיר את היחס החדש $\sim_k$ כך ש־ $x, y \in \Sigma^*$ מקיימות $x \not\sim_k y$ אם קיימת $z \in \Sigma^*$ עם $|z| \leq k$ המפרידה בין המילים. היחס $\sim_0$, למשל, מפריד בין מילים שמלכתחילה אחת מהן שייכת לשפה והשניה לא, כי המפריד היחיד זו המילה הריקה. עבור $\sim_0$ מחלקות השקילות הן $\{L, \overline{L}\}$.

עבור היחסים $\sim_1, \ldots, \sim_k$ יכולות להתווסף מחלקות שקילות. האלגוריתם יראה שהיחס $\sim$ המקורי מתקבל בתור $\sim_n$ סופי כלשהו.

- נתחיל ב־ $\{L, \overline{L}\} \leftarrow \sim_0$ ו־ $i \leftarrow 0$.
- לכל $p, q \in Q$
  - נאמר ש־ $p \sim_{i+1} q$ אלא אם כן:
  - $p \not\sim_i q$; <u>או</u>
  - קיים $\sigma \in \Sigma$ כך ש־ $\delta(q, \sigma) \neq \delta(p, \sigma)$.
- $i \leftarrow i + 1$ וחוזרים על התהליך עד שלא מתקבלים יותר שינויים במהלך הלולאה הפנימית.

אינטואיטיבית, מספר מחלקות השקילות הוא סופי וברור שהתהליך הזה עוצר בשלב כלשהו, ומתקבלת הקבוצה המינימלית של מחלקות שקילות. מהן אפשר לבנות אוטומט מצומצם לפי המצבים הנמצאים באותה מחלקת שקילות.

2.6 ביטויים רגולריים

הגדרה 2.25 ביטוי רגולרי מעל א"ב Σ הוא מהצורה:

1. "$\emptyset$" - עם השפה $L_\emptyset = \emptyset$;
2. "ε" - עם השפה $L_\varepsilon = \{\varepsilon\}$;
3. עבור $a \in \Sigma$, הביטוי "a" - עם השפה $L_a = \{a\}$;
4. אם R_1, R_2 ביטויים רגולריים מעל Σ, אז "$(R_1) \cup (R_2)$" הוא ביטוי רגולרי - עם השפה $L_{(R_1)\cup(R_2)} = L_{R_1} \cup L_{R_2}$;
5. אם R_1, R_2 ביטויים רגולריים מעל Σ, אז "$(R_1) \cdot (R_2)$" הוא ביטוי רגולרי - עם השפה $L_{(R_1)\cdot(R_2)} = L_{R_1} \cdot L_{R_2}$;
6. אם R ביטוי רגולרי מעל Σ, אז "$(R)^*$" ביטוי רגולרי - עם השפה $L_{(R)^*} = (L_R)^*$;
7. אם R ביטוי רגולרי מעל Σ, אז "$(R)^+$" מוגדר להיות כינוי לביטוי הרגולרי "$R(R)^*$".

<u>דוגמאות לביטויים רגולריים</u>:

- שפת המילים המכילות aba ומסתיימות ב־ b מתוארת ע"י $R = (a \cup b)^* aba (a \cup b)^* b$
- שפת כל המילים מעל $\{a, b\}$ מתוארת ע"י $R = (a \cup b)^*$
- שפת המילים שאינן מכילות שני אפסים רצופים, ואם אינן ריקות מתחילות באפס ומסתיימות באחד מתוארת ע"י $R = ((0(1)^+)^*$
- שפת המילים הלא ריקות באורך גדול מ־1 המתחילות ומסתיימות באותה אות מתוארת ע"י הביטוי $R = (a(a \cup b)^* a) \cup (b(a \cup b)^* b)$
- נשים לב גם ש־ $L(\text{"}\emptyset^*\text{"}) = L(\text{"}\varepsilon\text{"})$

משפט 2.26 יהי R ביטוי רגולרי מעל Σ. אזי $L_R \in REG$.

הוכחה: באינדוקציה שלמה על האורך של R.

עבור $|R| = 1$ הביטויים הרגולריים "a", "ε", "$\emptyset$" משרים כמובן שפות רגולריות.

עבור $|R| = n$ כאשר הוכחנו כבר עבור $< n$, מתבוננים בשלושת המקרים:

1. $R = (R_1) \cup (R_2)$ נובע מכך שאיחוד של שפות רגולריות הוא רגולרי, ולגבי R_1, R_2 משתמשים בהנחת האינדוקציה.
2. $R = (R_1) \cdot (R_2)$ נובע מכך ששרשור של שפות רגולריות הוא רגולרי.
3. $R = (R_1)^*$ נובע מכך שהסגור של קליני של שפה רגולרית הוא שפה רגולרית.

בזאת השלמנו את האינדוקציה. ■

הערה 2.27 מהמשפט מקבלים גם תהליך רקורסיבי לבניית NFA המזהה את הביטוי כולו מה־NFA־ים המזהים את חלקי הביטוי השונים.

משפט 2.28 לכל שפה רגולרית L קיים ביטוי רגולרי R כך ש־ $L = L_R$.

הוכחה: (לא פורמלית) נתבונן ב־DFA שמזהה את L, ונרצה "להיפטר" מכל המצבים פרט למצב ההתחלתי והמצב המקבל. כדי להיפטר ממצב ביניים מסוים, נכתוב על הקשת החדשה שמדלגת על המצב את הביטוי הרגולרי המתאים. ניתן להראות פורמלית שתהליך זה מביא לביטוי רגולרי שנשאר על הקשת האחת מהמצב ההתחלתי למצב המקבל. ■

הערה 2.29 המעבר בין שני הייצוגים של שפה - בתור אוטומט ובתור ביטוי רגולרי - עלול לקחת זמן אקספוננציאלי. (לא הוכחנו זאת.)

2.7 בעיות הכרעה לגבי שפות רגולריות

נשאל מספר שאלות לגבי שפות רגולריות, ובהמשך נחזור עליהן גם לגבי מודלים מורכבים יותר.

1. ריקנות: בתרגיל 2 ראינו כיצד אפשר לבדוק האם השפה של DFA מסוים היא ריקה - מחפשים מהלך בגרף מהמצב ההתחלתי אל המצב המקבל.

2. הכלה: בהינתן DFA־ים A, B כיצד נבדוק האם $L(A) \subseteq L(B)$?
השאלה שקולה לבדיקה $L(A) \cap \overline{L(B)} = \emptyset$. אנו יודעים למצוא את המשלים של אוטומט דטרמיניסטי באופן יעיל, ואת החיתוך באמצעות אוטומט מכפלה, והצטמצמנו בחזרה לבדיקת ריקנות.

3. אוניברסאליות: באופן דומה אפשר לבדוק האם $L(A) = \Sigma^*$ ע"י בדיקה האם $\overline{L(A)} = \emptyset$.

4. הבעיות הנ"ל לגבי NFA אינן כה טריוויאליות. כפי שראינו בתרגיל 11, ניתן לבנות NFA ששפתו היא ההשמות הלא־מספקות של נוסחת $3CNF$. כלומר, בדיקה האם $L(A) = \Sigma^*$ עבור NFA A מאפשרת לבדוק האם נוסחת $3CNF$ ניתנת לסיפוק, בעוד ש־ $3SAT \in NPC$.[1]

לעומת זאת, אנו יודעים לבדוק ריקנות של NFA בזמן פולינומי (ואף במקום לוגריתמי), באופן דומה ל־ DFA (חיפוש בגרף). ניתן להסיק שאנו לא יודעים למצוא משלים של NFA בזמן שאינו אקספוננציאלי.

3 שפות חסרות הקשר

שפות חסרות הקשר משמשות למשל לתיאור של שפות תכנות, והן חזקות יותר מהשפות הרגולריות. למשל, כבר השפה הפשוטה של הסוגריים המאוזנים אינה רגולרית (לפי למת הניפוח), אבל ניתנת לתיאור ע"י דקדוק חסר הקשר.

3.1 דקדוקים חסרי הקשר

הגדרה 3.1 דקדוק חסר הקשר (CFG) הוא רביעייה $G = (V, \Sigma, R, S)$ כאשר V קבוצה סופית של משתנים, Σ קבוצה סופית של טרמינלים ומתקיים $V \cap \Sigma = \emptyset$, R קבוצה סופית של כללי גזירה, ו־ S משתנה ההתחלה.
כלל גזירה הוא זוג סדור המכיל משתנה יחיד מצד שמאל שלו ומצד ימין הוא מכיל מחרוזת מעל הא"ב $\Sigma \cup V \cup \{\varepsilon\}$.

דוגמאות לדקדוקים חסרי הקשר:

1. שפת הסוגריים המאוזנים:

$$\begin{aligned} &\to A \\ A &\to (A) \\ A &\to AA \\ A &\to \varepsilon \end{aligned}$$

2. השפה $L = \{a^n b^m : m \geq n\}$:

$$\begin{aligned} &\to A \\ A &\to aAb \\ A &\to B \\ B &\to Bb \\ B &\to \varepsilon \end{aligned}$$

[1] ולמעשה, במופעים קודמים של הקורס אף הופיעה הוכחה שבדיקת אוניברסאליות של NFA היא שלמה ב־ PSPACE. לכן לא ניתן לצפות לפתרון יעיל של הבעיה אלא אם כן $P = \text{PSPACE}$.

3. שפת כל המילים מהצורה $a^i b^j c^k$ כאשר $i,j \geq 0$ ו־ $k \geq 1$:

$$\begin{array}{rcl} \to S & & \\ S & \to & ABC \\ A & \to & aA \\ A & \to & \varepsilon \\ B & \to & bB \\ B & \to & \varepsilon \\ C & \to & cC \\ C & \to & c \end{array}$$

הגדרה 3.2 השפה המתאימה לדקדוק חסר הקשר G תסומן $L(G)$ ומכילה בדיוק את המילים מעל Σ המתקבלות כאשר מתחילים במחרוזת S ובכל שלב מפעילים כלל גזירה ע"י החלפת המשתנה שבו במחרוזת שבצד ימין שלו, עד שנשארים עם טרמינלים בלבד.
שפה המתאימה לדקדוק חסר הקשר תיקרא שפה חסרת הקשר (CFL). קבוצת השפות חסרות ההקשר תסומן CFL.

הערה 3.3 ניתן לדבר גם על עץ הגזירה של מילה מסוימת בשפה, שבו טרמינלים ומשתנים:

משפט 3.4 $REG \subset CFL$

הוכחה: תהי $L \in REG$ ו־ A DFA שמקבל אותה, $A = (Q, \Sigma, \delta, q_0, F)$. נבנה G CFG עם אותה השפה. נסמן $G = (Q, \Sigma, R, q_0)$ ונגדיר ב־ R את כללי הגזירה:
לכל $a \in \Sigma,\ q_1, q_2 \in Q$ שעבורם $\delta(q_1, a) = q_2$ יהיה כלל גזירה מהצורה $q_1 \to aq_2$. בנוסף יהיה כלל $q \to \varepsilon$ לכל $q \in F$.
נשאר להראות ש־ $L(A) = L(G)$. הטענה העיקרית היא שאחרי t צעדי גזירה נקבל מחרוזת מהצורה wq כאשר $q = \delta^*(q_0, w)$, או מחרוזת מהצורה w כאשר $w \in L(A)$. את זה אפשר להראות באינדוקציה על מספר צעדי הגזירה, ולכן $L(G) \subseteq L(A)$.
להיפך, לכל $w \in \Sigma^*$ המילה wq עבור $q = \delta^*(q_0, w)$ נגזרת מ־ G, ושוב מראים זאת באינדוקציה על אורך המילה. ■

הערה 3.5 דקדוק חסר הקשר נקרא דקדוק שמאלי אם כל כלל גזירה הוא מהצורות $A \to \varepsilon$ או $A \to wB$ כאשר w מכילה רק טרמינלים ו־ B משתנה. קל לראות שדקדוק שמאלי נותן שפות רגולריות בלבד. (תרגיל 4, שאלה 5.)

הגדרה 3.6 דקדוק חסר הקשר הוא חד־משמעי אם כל מילה ניתנת לגזירה בדרך אחת בלבד. למשל, הדקדוק הבא אינו חד משמעי,

$$\begin{array}{rcl} \to A & & \\ A & \to & AA|(A)|\varepsilon \end{array}$$

משום שניתן לגזור ממנו את המילה () במספר דרכים. למשל: $() \to (A) \to ()$ היא דרך אחת, ו־ $A \to AA \to (A) \to ()$ היא דרך שניה.

הערה 3.7 לא לכל שפה חסרת הקשר יש דקדוק חסר הקשר חד־משמעי שמתאר אותה. (לא ראינו מדוע.)

3.2 פעולות על שפות חסרות הקשר

בהנתן L, M שפות חסרות הקשר מעל א"ב Σ, נרצה לבדוק האם $L \cap M,\ L \cup M,\ L \cdot M,\ \overline{L},\ L^*$ הן חסרות הקשר, בדומה למה שעשינו עבור שפות רגולריות.

טענה 3.8 $L, M \in CFL \implies L_1 \cup L_2 \in CFL$

הוכחה: ראשית נשנה את שמות המשתנים כדי שמשתני L_1, L_2 יהיו קבוצות זרות. ואז, אם S_1, S_2 המשתנים ההתחלתיים אז ניצור משתנה חדש S שיהיה המשתנה ההתחלתי של $L_1 \cup L_2$ וכללי גזירה חדשים $S \to S_1$ ו־ $S \to S_2$. קל לראות שהדקדוק המתקבל הוא הדרוש. ■

הערה 3.9 לעומת זאת, שפות חסרות הקשר אינן סגורות לחיתוך (ולכן גם לא למשלים). למשל, נראה בהמשך שהשפה $\{a^n b^n c^n\}$ אינה חסרת הקשר, אבל היא מתקבלת בתור $\{a^l b^n c^n\} \cap \{a^n b^n c^l\}$ שהן שתיהן שפות חסרות הקשר. למשל, להלן הדקדוק עבור השפה $\{a^n b^n c^l\}$:

$$\begin{aligned} &\to S \\ A &\to aAb|\varepsilon \\ C &\to cC|\varepsilon \end{aligned}$$

ובאופן דומה אפשר למצוא דקדוק חסר הקשר ל־ $\{a^l b^n c^n\}$.

טענה 3.10 $L, M \in CFL \implies L \cdot M \in CFL$

הוכחה: בדומה להוכחה המתאימה לשפות רגולריות, כאשר משתמשים באוטומט מחסנית. ■

טענה 3.11 $L \in CFL \implies L^* \in CFL$

הוכחה: בדומה להוכחה המתאימה לשפות רגולריות, כאשר משתמשים באוטומט מחסנית. ■

טענה 3.12 $L \in REG, M \in CFL \implies L \cap M \in CFL$

הוכחה: (תרגיל 5, שאלה 1) סקיצת ההוכחה: בניית מכפלה של DFA שמזהה את L ו־ PDA שמזהה את M. כיוון שה־ DFA לא צריך מחסנית, אין "התנגשות". ■

3.3 צורה נורמלית של חומסקי

אנו מחפשים צורה פשוטה לתאר דקדוקים כדי שיהיה לנו יותר קל להוכיח דברים על שפות חסרות הקשר. אם נראה שכל דקדוק חסר הקשר ניתן להצגה בצורה פשוטה יחסית שגם חוסמת בקלות את גודל המילה המתקבלת משרשרת גזירה חסומה, זה יקל עלינו מאוד.

הגדרה 3.13 נאמר שדקדוק חסר הקשר הוא בצורה נורמלית של חומסקי (Chomsky Normal Form) אם כל כללי הגזירה שלו הם מהצורות הבאות:

1. $S \to \varepsilon$ ואין חוקים נוספים מהצורה $A \to \varepsilon$
2. $A \to BC$ כאשר B, C משתנים שאינם המשתנה ההתחלתי אבל יכולים להיות זהים או שווים ל־ A
3. $A \to \sigma$ כאשר $\sigma \in \Sigma$ (טרמינל)

טענה 3.14 לכל דקדוק חסר הקשר קיים דקדוק חסר הקשר בעל אותה שפה שהוא בצורה נורמלית של חומסקי.

הוכחה: נראה אלגוריתם להפיכת CFG לצורה נורמלית של חומסקי:

1. נוסיף משתנה התחלתי חדש S_0 וכלל גזירה $S_0 \to S$. קל לראות שהשפה לא משתנה כתוצאה מזה, אלא רק הארכנו כל גזירה של מילה בעוד שלב אחד.
2. לכל כלל גזירה מהצורה $A \to \varepsilon$ (כאשר $A \neq S_0$) נעבור על כל הכללים מהצורה $B \to wAu$ ונוסיף להם את החוק $B \to wu$ (כאשר w, u מחרוזות כלשהן)[2]. את הכלל $A \to \varepsilon$ נוכל להסיר כעת, וגם זה לא פוגע בשפה.
3. לכל כלל מהצורה $A \to B$ (כאשר $A \neq B$[3]) נעבור על כללים מהצורה $C \to wAu$ ונוסיף את הכלל $C \to wBu$. לבסוף נוכל להסיר את הכלל $A \to B$ וגם זה לא פוגע בשפה. את זאת נעשה גם עבור כללים שבהם $B \in \Sigma$.
4. לכל כלל מהצורה $A \to v_1 \dots v_k$ כאשר $k > 2$ נוסיף משתנים חדשים $u_2, \dots, u_{k-1}$ ואת כללי הגזירה החדשים $A \to v_1 u_2$, $u_2 \to v_2 u_3$, $u_3 \to v_3 u_4$ וכו' עד $u_{k-1} \to v_{k-1} v_k$. כמובן, לא פגענו בשפה.
5. נוסיף משתנים מיוחדים X_σ לכל $\sigma \in \Sigma$ ואת הכללים $X_\sigma \to \sigma$. בכל כלל שבו באגף ימין יש σ נחליף אותו ב־ X_σ.

נותר להוכיח (לא עשינו זאת) שהתהליך מסתיים. את זה שהשפה לא משתנה קל לראות. ■

הערה 3.15 כתוצאת לוואי, אנו מגלים האם דקדוק גוזר את המילה הריקה. אם קיבלנו את הכלל $S_0 \to \varepsilon$ אז כן, אחרת לא.

[2] לדוגמה, אם היה הכלל $B \to ACAD$ אז נצטרך להוסיף את הכללים $B \to ACD$, $B \to CAD$ ו־ $B \to CD$.
[3] שכן כללים מהצורה $A \to A$ אפשר להסיר מהדקדוק ללא פגיעה בשפה.

<u>דוגמת הרצה של האלגוריתם</u>:
נתחיל מהדקדוק הבא:

$$\begin{aligned} \to S & \\ S &\to ASA|aB \\ A &\to B|S \\ B &\to b|\varepsilon \end{aligned}$$

הכלל ה"תקין" היחיד הוא $B \to b$. נתחיל את התהליך.
1. ניצור כלל $S_0 \to S$ חדש.
2. נתקן את $B \to \varepsilon$ ע"י הסרתו על חשבון החוקים $A \to \varepsilon$ ו־ $S \to a$.
3. נתקן את הכלל $A \to \varepsilon$ ע"י הסרתו על חשבון החוקים $S \to AS$ ו־ $S \to SA$.
4. נתקן את הכלל $A \to B$ ע"י חסרתו על חשבון החוקים $S \to BS$, $S \to SB$, $S \to BSB$, $S \to BSA$, $S \to ASB$.
אנו רחוקים מסיום אך נעצור כאן.

3.4 למת הניפוח לשפות חסרות הקשר

גם עבור שפות חסרות הקשר למת הניפוח עובדת, אבל יהיה קצת יותר קשה להראות מדוע. הצורה הנורמלית של חומסקי מאוד תעזור לנו כדי לתאר היטב את עץ הגזירה של הדקדוק דנן.

למה 3.16 תהי $L \in CFL$. אז

$$\exists p>0 \quad \forall w \in L, |w|>p \quad w=uxyzv,\ |xz|>0,\ |xyz| \le p \qquad \forall i \in \mathbb{N} \cup \{0\} \quad ux^iyz^iv \in L$$

הוכחה: תהי $L \subseteq \Sigma^*$ חסרת הקשר ויהי $G=(Q,\Sigma,R,S)$ דקדוק חסר הקשר שגוזר אותה. יהי n האורך המקסימלי של מחרוזת המופיעה בכלל גזירה מתוך R.
נגדיר את קבוע הניפוח להיות $p := n^{|Q|+1}$ ונראה שהוא מקיים את הדרוש. תהי $w \in L$ עם $|w| > p$. יהי T עץ גזירה מינימלי הגוזר את w ע"פ G (מינימלי מבחינת כמות הצמתים בעץ).
הגובה של T: אם העץ בגובה l אז מספר העלים הוא לכל היותר n^l (בגלל החסם על אורך כלל גזירה). לכן הגובה של T הוא גדול מ־ $|Q|+1$, כי המילה באורך p וכל אות בה היא עלה בעץ הגזירה.
לכן, יש מסלול $S_0, S_1, \ldots, S_l = S$ באורך $|Q| <$ המתחיל בעלה ומגיע לשורש של T. לפי עיקרון שובך היונים, חייב להיות איזה אינדקס ראשון i כך שקיים אינדקס $i > j$ המקיים $S_i = S_j$, כלומר i הוא המקום הראשון שבו חוזרים על משתנה שכבר ראינו במהלך המסלול.
נסמן ב־ T_i את תת־העץ שתלוי על S_i וב־ T_j את תת־העץ התלוי על S_j. כעת ידועה לנו החלוקה של w מהלמה. כל הטרמינלים משמאל ל־ T_i יהיו u ואלה שמימין לו יהיו v. בתוך T_i כל הטרמינלים משמאל ל־ T_j יהיו x, מימין ל־ T_j יהיו z, ובתוך T_j יהיו y.
עתה נפנה לטענות הלמה:
1. הגובה של T_i הוא $|Q|+1 \ge$ ולכן מספר העלים בו $p = n^{|Q|+1} \ge |xyz|$. כלומר, כפי שרצינו.
2. נניח בשלילה ש־ $xz = \varepsilon$, אז אם נחליף את T_i ב־ T_j נקבל עץ גזירה חוקי ע"פ G שעדיין גוזר את w ולבטח יש בו פחות צמתים (לפחות חסכנו את הצומת S_i), וזו סתירה. מותר לנו לעשות את ההחלפה הזאת כי $S_i = S_j$ והצומת הקודם ל־ S_i השתמש בכלל גזירה שבו אותו משתנה. לכן $|xz| > 0$.
3. על מנת לנפח ב־ $i = 0$ עלינו להחליף את T_i ב־ T_j, בדיוק כמו ב־ (2). קל לראות שזה מספק מילה בשפה מאותם שיקולים.
4. על מנת לנפח ב־ $i = 2$ עלינו להדביק במקום T_j עותק של T_i (עם T_j בפנים). קל להשתכנע שזה עובד גם עבור $i > 2$ (אינדוקציה). ■

<u>דוגמאות לשימוש בלמה</u>:
1. נתבונן בשפה הבאה מעל $\Sigma = \{a, b, \#\}$:

$$L = \{w\#u : \ w, u \in \{a,b\}^* \wedge \exists x, y : \ w = xuy\}$$

נוכיח שהיא אינה חסרת הקשר בעזרת למת הניפוח. נניח בשלילה שהיא חסרת הקשר. מהלמה קיים p כך שאפשר לפרק את המילה $w = a^pb^p\#a^pb^p \in L$ ל־ $w = uvxyz$ לפי תנאי הלמה. אז $|vxy| \le p$ ונחלק למקרים:

1. vxy נמצא משמאל ל־ $\#$. במקרה זה אפשר לנפח את vy באמצעות $i = 0$ וזה מקטין את הביטוי $a^p b^p$, מה שגורם לזה שהמילה כבר לא בשפה, כי הביטוי מימין ל־ $\#$ כבר לא יכול להיות תת־מחרוזת של ביטוי קצר יותר משמאל.

2. vxy נמצא מימין ל־ $\#$. במקרה זה אפשר לנפח באמצעות $i = 2$ וזה שוב גורם לצד ימין להיות ארוך יותר מצד שמאל, ולכן הוא לא יכול להיות תת־מחרוזת.

3. vxy מכיל את $\#$. גם כאן נחלק למקרים:

   (א) v מכיל $\#$. אפשר לנפח עם $i = 0$ ולקבל מילה שאין בה בכלל $\#$ ולכן כמובן אינה בשפה.

   (ב) y מכיל $\#$. כנ"ל.

   (ג) x מכיל $\#$. זה אומר ש־ v מורכב רק מ־ b־ים ו־ y מורכב רק מ־ a־ים. שוב נחלק למקרים:

   i. $|v| > 0$ אז ננפח ב־ $i = 0$ ואז בצד שמאל הקטנו את מספר ה־ b־ים ובצד ימין לא, ולכן לא יכול להיות שצד ימין הוא תת מחרוזת של צד שמאל.

   ii. $|y| > 0$ אז ננפח ב־ $i = 2$ ואז הגדלנו את מספר ה־ a־ים מימין ולא שינינו את מספרם משמאל ושוב המילה אינה בשפה.

בזאת השלמנו את ההוכחה ש־ L אינה חסרת הקשר.

2. נוכיח ש־ $L = \{a^n b^n c^n : n \in \mathbb{N}\} \notin CFL$. נניח בשלילה ש־ $L \in CFL$ ויהי p קבוע הניפוח שלה. נעיין במילה $w = a^{5p} b^{5p} c^{5p}$. ע"פ למת הניפוח אפשר לחלק אותה ל־ $w = uxyzv$. גם ב־ x וגם ב־ z יופיעו אותיות מסוג אחד בלבד, ואז הניפוח (למשל עם $i = 2$) ייצור מצב שמספר מופעי אות אחת לפחות יגדל ומספר מופעי אות אחת לפחות לא ישתנה. המילה המתקבלת אינה ב־ L, בסתירה.

3.5 בעיות הכרעה לגבי שפות חסרות הקשר

1. נתחיל מבעיית ה־ membership - האם מילה w נגזרת מהדקדוק G?
תחילה נשים לב שמספיק להסתכל על דקדוקים בצורה נורמלית של חומסקי.
נתאר אלגוריתם תכנון דינאמי לבעיה. נחזיק טבלה $n \times n$ עבור המילה $w = \sigma_1 \dots \sigma_n$, כאשר בתא ה־ i, j יהיו כל המשתנים של G שמהם יש גזירה של $\sigma_i \dots \sigma_j$ באמצעות כללי הגזירה של G.
מקרה הבסיס הוא כאשר $i = j$ ומחפשים את המשתנים שמהם אפשר לגזור את σ_i; שאר המקרים מתקבלים בצורה רקורסיבית: עבור $j > i$ המשתנים X שמהם אפשר לגזור את $\sigma_i \dots \sigma_j$ הם כאלה שעבורם יש כלל מהצורה $X \to YZ$ ופירוק $\sigma_i \dots \sigma_j = \sigma_i \dots \sigma_k \sigma_{k+1} \dots \sigma_j$ כך שרקורסיבית אפשר לגזור מ־ Y את $\sigma_i \dots \sigma_k$ ומ־ Z את $\sigma_{k+1} \dots \sigma_j$.
S מופיע בתא ה־ $1, n$ של הטבלה אסם אפשר לגזור את המילה w מ־ G.

2. באופן די דומה אפשר לפתור גם את בעיית הריקנות.
נתחיל בכך ש"נסמן" את כל הטרמינלים. לכל כלל גזירה מהצורה $A \to X_1 \dots X_k$ נבדוק האם $X_1, \dots, X_k$ כולם מסומנים. אם כן, נסמן גם את A. נמשיך באופן כזה עד שלא מסמנים יותר שום דבר.
אם S סומן, השפה של הדקדוק אינה ריקה; אחרת, היא ריקה. (הוכחת נכונות וזמן ריצה - שהוא פולינומיאלי באורך הייצוג של הדקדוק - בסיכום תרגול 4.)
דרך אחרת היא להעביר את הדקדוק לצורה נורמלית של חומסקי, ולבדוק האם $S_0 \to \varepsilon$ זהו כלל בדקדוק.

3. בהמשך נראה שלא ניתן להכריע את בעיית ה־ ALL_{CFG} - כלומר לקבוע האם דקדוק מסוים מקבל את כל המילים.

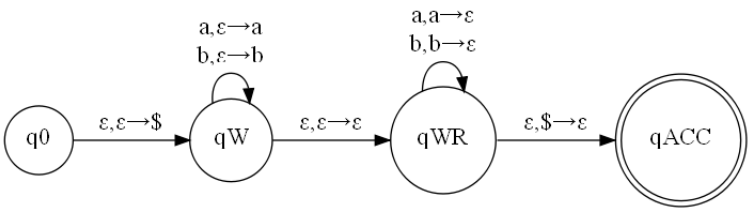
3.6 אוטומט מחסנית

אוטומט מחסנית (Pushdown Automaton ־ PDA) הוא אוטומט לא דטרמיניסטי עם זיכרון עזר בצורת מחסנית. נשים לב שמילת הקלט אינה מופיעה במחסנית (בניגוד למכונות טיורינג שנראה בהמשך). נשתמש בו לזיהוי שפות חסרות הקשר.

הגדרה 3.17 אוטומט מחסנית הוא שישייה סדורה $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ כאשר Q קבוצת מצבים סופית, Γ א"ב המחסנית, Σ א"ב השפה, $q_0 \in Q$ מצב התחלה, $F \subseteq Q$ קבוצת המצבים המקבלים, ו־ δ פונקציית המעברים $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\}$.

הערה 3.18 בגלל הגדרה זו, אין לנו צורך במספר מצבי התחלה. בניגוד ל־ NFA, כאן אנו "מרפדים" את המילה ב־ ε־ים, כלומר החישוב יכול להתחיל במעבר ε.

דוגמא לאוטומט מחסנית:
אוטומט המחסנית הבא מזהה את השפה $L = \{ww^R\}$ מעל הא"ב $\Sigma = \{a, b\}$:



בציורים של אוטומט מחסנית, כשנכתוב $a, b \to c$ (כאשר $a \in \Sigma \cup \{\varepsilon\},\ b, c \in \Gamma \cup \{\varepsilon\}$) הכוונה היא שכאשר האוטומט קורא את האות a מהקלט ושולף מהמחסנית את האות b, הוא עובר למצב הבא ודוחף את האות c למחסנית.

הגדרה 3.19 קונפיגורציה של PDA מורכבת מהמצב שבו הוא נמצא, וגם מתוכן המחסנית. הקונבנציה שלנו תהיה לכתוב את תחתית המחסנית מצד ימין ואת ראשה בצד שמאל. למשל, האוטומט מהדוגמה הקודמת לאחר קריאת המילה ab יהיה בקונפיגורציה $(q_w, ba\$)$.

צעד חישוב הוא מעבר חוקי בין הקונפיגורציות. צעד חישוב חוקי על האות α הוא מעבר $(q_1, s_1) \to (q_2, s_2)$ כאשר $s_1 = as$ ו־ $s_2 = bs$, $a, b \in \Gamma \cup \{\varepsilon\}$, $(q_2, b) \in \delta(q_1, \alpha, a)$.

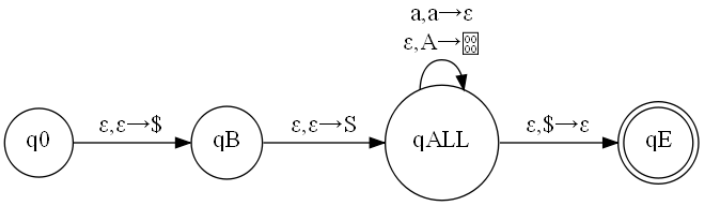
חישוב מקבל הוא חישוב חוקי המסתיים במצב מקבל, ושפת האוטומט היא אוסף המילים שעבורן קיים חישוב מקבל. (ולהיפך - מילה אינה מתקבלת ע"י האוטומט אסם כל חישוב שלו עליה לא מקבל.)

הערה 3.20 בהחלט ייתכן שנרצה בצעד מסוים להוציא אות מסוימת ולהכניס למחסנית אותה + אות נוספת. במודל שלנו, נצטרך לעשות את זה באמצעות שני מצבים - בראשון נוציא את האות ונחזיר אותה, ובשני נכניס את האות הנוספת שרצינו.

משפט 3.21 יהי G דקדוק חסר הקשר. אזי קיים A אוטומט מחסנית כך ש־ $L(A) = L(G)$.

הוכחה: (בערך) אפשר להכליל אוטומט מחסנית (ע"י הוספה של הרבה מצבים) כדי שנוכל בכל שלב לקרוא מילה (ב־ Γ^*) מהמחסנית ולכתוב מילה למחסנית.

אם כך, נוכל לבנות את האוטומט הבא עם Σ הטרמינלים של הדקדוק:



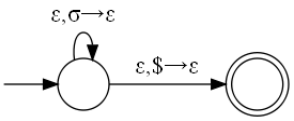
כאשר הקשת $\varepsilon, A \to \square$ משוכפלת לכל כלל גזירה מהצורה $A \to \square$ והקשת $a, a \to \varepsilon$ משוכפלת לכל טרמינל $a \in \Sigma$.

כעת, אם מוציאים מהמחסנית משתנה אז אנחנו "מנסים לגזור אותו" ואם מוציאים טרמינל מהמחסנית אז משווים אותו לאות הנוכחית במילה, וממשיכים. ■

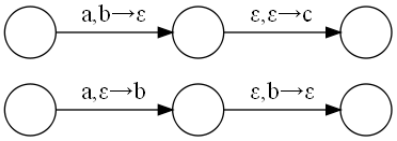
משפט 3.22 יהי A אוטומט מחסנית. אזי קיים דקדוק חסר הקשר G כך ש־ $L(G) = L(A)$.

הוכחה: נניח בה"כ ש:

0. בתחילת הריצה A דוחף $ למחסנית.
1. ל־ A מצב מקבל יחיד. זה בה"כ כי אחרת נוסיף מצב מקבל וקשתות $\varepsilon, \varepsilon \to \varepsilon$ מהמצבים המקבלים הקודמים למצב המקבל החדש.
2. A מקבל רק אם המחסנית ריקה. זה בה"כ כי אחרת נוסיף מצב מקבל חדש ומהמצב המקבל הישן נוסיף קשתות עצמיות $\varepsilon, \sigma \to \varepsilon$ לכל $\sigma \in \Gamma$, ועוד מעבר $\varepsilon, \$ \to \varepsilon$ למצב המקבל החדש.



3. בכל מעבר A דוחף אות למחסנית או מוציא אות מהמחסנית, אך לא שניהם. זה בה"כ - נטפל בשני המקרים:
   עבור מעבר $a, \varepsilon \to b$ נבנה מצב נוסף ומעברים $a, \varepsilon \to b$ ו־ $\varepsilon, b \to \varepsilon$.
   עבור מעבר $a, b \to c$ נבנה מצב נוסף ומעברים $\varepsilon, a \to b$ ו־ $a, b \to \varepsilon$.



נשים לב של־ A יש ריצה מקבלת על w אסם ל־ A יש ריצה מ־ q_0 ל־ q_{acc} (המצב המקבל) כך שהמחסנית ריקה בתחילת הריצה ובסופה.

כעת לכל $p,q \in Q$ יהיה משתנה A_{pq} כך שלכל מילה $x \in \Sigma^*$ מתקיים ש־ $A_{pq} \rightsquigarrow x$ אסם יש ל־ A ריצה מ־ p ל־ q שמשמרת את ריקנות המחסנית. $(*)$

נבחין בין שני מצבים:

1. במהלך הריצה מ־ p ל־ q המחסנית מתרוקנת במצב r. אם כך, החלק של w שעברנו עליו בין p ל־ r נגזר מ־ A_{pr} תוך שמירה על ריקנות המחסנית, והחלק של w שעברנו עליו בין r ל־ q נגזר מ־ A_{rq} תוך שמירה על ריקנות המחסנית.
2. במהלך הריצה כנ"ל המחסנית מתרוקנת רק בהגיענו למצב q. אם כך, נתבונן על המצב r שהוא המצב הראשון בדרך מ־ p ל־ q ועל המצב s שהוא המצב האחרון לפני q בדרך הנ"ל. במקרה זה, האות שנשלפת בין s ל־ q היא אותה אות שנדחפה בין p ל־ r - לפי ההנחות שלנו.

מכאן מוגדרים החוקים:

1. לכל $r,p,q \in Q$ נגדיר את הכלל $A_{pq} \to A_{pr}A_{rq}$
2. לכל $p,q,r,s \in Q$ ולכל $\gamma \in \Gamma,\ a,b \in \Sigma \cup \{\varepsilon\}$ נגדיר את הכלל $A_{pq} \to aA_{rs}b$ אם מתקיים $(r,\gamma) \in \delta(p,a,\varepsilon)$ ו־ $(q,\varepsilon) \in \delta(s,b,\gamma)$.
3. לכל $p \in Q$ נגדיר את הכלל $A_{pp} \to \varepsilon$.

בנוסף, המשתנה ההתחלתי יהיה $A_{q_0,q_{acc}}$.

עכשיו יש להוכיח את $(*)$ - זה נעשה באינדוקציה. (ר' סיכום תרגול 5 לעוד פרטים.) ■

חלק II
חישוביות

4 מכונות טיורינג

בפרק זה נתבונן במכונות טיורינג, השקולות למחשב מבחינת יכולת חישוב.

4.1 מכונות טיורינג "פשוטות"

למעשה, מכונת טיורינג היא אוטומט סופי דטרמיניסטי המצויד גם בסרט חישוב דו־כיווני (רשימה מקושרת דו־כיוונית). לראש הקריאה של האוטומט יש אפשרות לנוע ימינה ושמאל על הסרט, שהוא אמנם סופי אבל לא חסום. הקלט של המכונה מגיע על הסרט ונוכל לקרוא אותו כמה פעמים שנרצה ולכתוב מידע חדש על הסרט כרצוננו.

הגדרה 4.1 מכונת טיורינג היא שביעייה סדורה $M = (Q,\Sigma,\Gamma,q_0,q_{rej},q_{acc},\delta)$ כאשר Q קבוצת המצבים, Σ א"ב הקלט, Γ א"ב הסרט כאשר $\Sigma \subseteq \Gamma$ ו־ $\sqcup \in \Gamma \setminus \Sigma$ תו מיוחד, $q_0 \in Q$ המצב ההתחלתי, $q_{rej} \in Q$ המצב הדוחה, $q_{acc} \in Q$ המצב המקבל, ו־ δ פונקציית המעברים $\delta : Q \times \Gamma \to Q \times \Gamma \setminus \{\sqcup\} \times \{R,L\}$. (המשמעות של R,L היא "ללכת ימינה" או "ללכת שמאלה".)

קונפיגורציה של מכונת טיורינג היא (u,q,v) כאשר $q \in Q$, $u,v \in \Gamma^*$ מתארים את מה שמשמאל לראש הקורא/כותב ומה שמימין לו.

נסמן $v = av'$ כאשר $a \in \Gamma \setminus \{\sqcup\}$ אם $v \neq \varepsilon$ או $a = \sqcup$ ו־ $v' = \varepsilon$ אחרת. נניח ש־ $\delta(q,a) = (q',b,R)$. במקרה זה הקונפיגורציה העוקבת של (u,q,v) היא (ub,q',v'). כנ"ל עבור L.

ריצה חוקית של מכונת טיורינג היא סדרה (שאינה בהכרח סופית) של קונפיגורציות עוקבות. הריצה מסתיימת אם הסדרה סופית והמצב בקונפיגורציה האחרונה הוא q_{acc} או q_{rej}.

המכונה מקבלת את המילה (המילה בשפה של המכונה) אם יש ריצה שהקונפיגורציה ההתחלתית שלה היא (ε,q_0,w) והקונפיגורציה הסופית שלה היא ב־ q_{acc}.

המכונה מכריעה את השפה L אם היא תמיד עוצרת, והיא עוצרת במצב מקבל רק על קלטים $w \in L$.

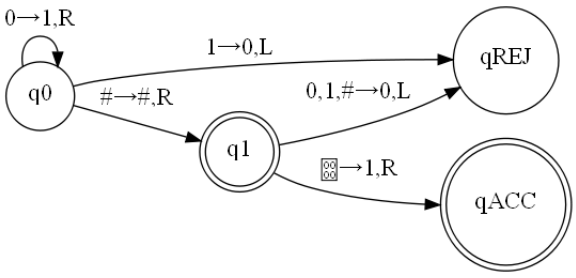
המכונה מזהה את השפה L אם היא עוצרת ומקבלת על קלטים $w \in L$, ועל קלטים $w \notin L$ היא עוצרת ודוחה או שאינה עוצרת.

הערה 4.2 1. השפה של הפולינומים $p(x,y,z)$ ומספרים a,b,c כך ש־ $p(a,b,c) = 0$ ניתנת להכרעה ע"י מכונת טיורינג.

2. השפה של הפולינומים במקדמים שלמים $p(x_1,\ldots,x_k)$ כך שקיימים $a_1,\ldots,a_k$ שלמים המאפסים את הפולינום ניתנת לזיהוי ע"י מכונת טיורינג (ניתן לעבור על כל האפשרויות) אבל אינה ניתנת להכרעה. זוהי בעיה מספר 10 של הילברט.
3. (ללא קשר) ניתן לגרום למכונה להתמודד עם שפות אינסופיות ע"י החלפת q_{rej} ב־ q_{loop}. (לא פיתחנו עוד את הרעיון הזה.)
4. על פני אוטומט רגיל, הוספנו את האפשרות לכתוב ולזוז שמאלה וימינה על הסרט. כל אחד משני הפיצ'רים האלה לא מספק עדיין את הכוח של מכונת טיורינג אלא נשאר בתחום השפות הרגולריות.
5. מכונת טיורינג יכולה לקבל, לדחות, או "להיתקע" - כלומר לעולם לא לעצור. בניגוד לאוטומט, מכונת טיורינג יכולה לא לעצור ובכל זאת לא לחזור על אותה קונפיגורציה פעמיים (שכן הסרט אינו חסום).

<u>דוגמאות למכונות טיורינג</u>:

1. להלן מכונת טיורינג בעלת השפה $L = \{0^*\#\}$ מעל הא"ב $\Sigma = \{0, 1, \#\}$:

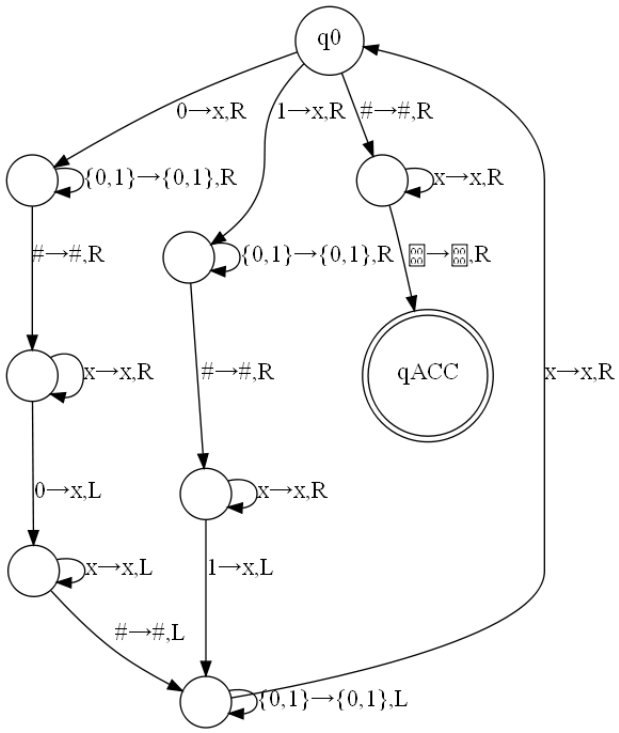


2. השפה $L = \{w\#w : \ w \in \{0,1\}^*\}$ מעל הא"ב $\{0, 1, \#\}$ אינה חסרת הקשר. ניתן להראות זאת ישירות באמצעות למת הניפוח לשפות חסרות הקשר.

אבל, אפשר לזהות אותה באמצעות מכונת טיורינג. תחילה, נתאר אלגוריתם ואז נבנה את המכונה ממש (מה שלא נעשה בדרך כלל, כי זה מגעיל וארוך):

1. החלף את האות מתחת לראש ב־ x וזכור אותה
2. לך ימינה עד $\#$
3. לך ימינה צעד אחד, וימינה כל עוד יש x
4. אם האות מתחת לראש היא כמו שזכרת, סמן x ולך שמאלה עד $\#$
5. לך שמאלה עד x, ולך ימינה צעד אחד
6. אם האות היא לא $\#$, חזור ל־ (1)
7. לך ימינה על x־ים, אם הגעת ל־ $\sqcup$, קבל

להלן ציור של המכונה, כאשר לא ציירנו את הקשתות שהולכות למצב הדוחה q_{rej}:



3. נבנה מכונת טיורינג שבודקת האם גרף לא מכוון הוא קשיר. נשתמש ב־ $\langle\rangle$ כדי לסמן קידוד של משהו, למשל $\langle G\rangle$. קידוד סביר של גרף הוא רשימה של הקודקודים בתור מספרים בינאריים מופרדים ע"י $\#$, שאחריהם רשימה של הצלעות בתור מספרים בינאריים. לדוגמא:

$$\langle G\rangle = 000\#001\#010\#\ldots\#\#\#(000\#001)(010\#000)\ldots$$

פיסת הקידוד הנ"ל מתארת את הקודקודים v_0, v_1v_2 והצלעות $(v_0, v_1), (v_2, v_0)$.
המכונה תשתמש בשני סרטים (בהמשך נראה שזה שקול למכונה עם סרט אחד) ותפעל כך:
1. בדוק שהקלט מהווה קידוד חוקי של גרף.
2. סמן את הקודקוד הראשון של G ע"י העברתו לסרט של הקודקודים שכבר סומנו.
3. חזור עד שלא מסומנים קודקודים חדשים: לכל קודקוד ב־ G, סמן אותו אסם יש קשת בינו לבין קודקוד שכבר סומן.
4. עבור על כל קודקודי G. אם כולם מסומנים, עצור וקבל; אחרת, עצור ודחה.

הערה 4.3 1. ניתן לתאר אלגוריתם באמצעות מכונת טיורינג בכמה מישורים - התיאור הפורמלי באמצעות שביעייה, תיאור ברמת המימוש (האופן שבו המכונה פועלת), ותיאור בשפה טבעית של האלגוריתם. בדרך כלל נפנה לאפשרות האחרונה.
2. כאשר אנו מדברים על מכונות טיורינג שמקבלות קידוד של "משהו" כקלט, בהחלט ייתכן שלא כל גיבוב תווים מעל הא"ב של המכונה יהיה קלט חוקי, כלומר קידוד חוקי של "משהו". מקרים כאלה לא יעניינו אותנו במיוחד, כי בדרך כלל אפשר בזמן קצר מאוד לוודא בתחילת הריצה שאכן מדובר בקלט לא חוקי, ולדחות אותו (או לעשות כל דבר אחר שנרצה).

4.2 וריאציות על מכונות טיורינג

הגדרה 4.4 מכונת טיורינג עם שני סרטים מוגדרת כך - המילה כתובה על הסרט הראשון, והמכונה יכולה להשתמש בשני הסרטים לצורך עבודה. פונקציית המעברים היא $\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{R, L\}^2$ מכיוון שיש שני ראשים קוראים/כותבים שהמכונה משתמשת בהם.

טענה 4.5 לכל מכונת טיורינג עם שני סרטים יש מכונת טיורינג שקולה (בעלת אותה השפה - גם מבחינת קבלה ודחייה, וגם מבחינת עצירה).

הוכחה: נעבוד עם א"ב מחסנית של רביעיות, $\Gamma' = \Gamma \times \Gamma \times \{0, 1\} \times \{0, 1\}$ כאשר אות כמו $(a, b, 1, 0)$ משמעותה שעל הסרט הראשון כתוב a, על הסרט השני כתוב b, הראש של הסרט הראשון נמצא מעל האות והראש השני של המכונה לא נמצא מעל האות.
נגדיר $M' = (Q', \Sigma, \Gamma', \delta', q'_0, q'_{acc}, q'_{rej})$. המכונה פועלת בשני שלבים:
1. נעבור על כל הסרט ונהפוך כל אות $\sigma \in \Sigma$ לאות $(\sigma, \sqcup, 0, 0)$ ואת האות השמאלית ביותר ל־ $(\sigma, \sqcup, 1, 1)$ - שכן שני הראשים מתחילים באותו המקום.
2. נרוץ על הסרט ונחפש אות מהצורה $(\square, \square, 1, \square)$ - מה שנמצא מתחת לראש של הסרט הראשון. נזכור מה האות בסרט הראשון. נמשיך ימינה עד הסוף.
3. נחזור שמאלה ונחפש אות מהצורה $(\square, \square, \square, 1)$ - מה שנמצא מתחת לראש של הסרט השני. נזכור מה האות בסרט השני. נחזור שמאלה עד $\sqcup$.
4. נרוץ ימינה ונחפש אות מהצורה $(\sqcup, \sqcup, 1, \sqcup)$, נבצע סימולציה של פעולת M על הסרט הראשון. כנ"ל לסרט השני.
5. אם M אומרת לעבור ל־ q_{acc} או q_{rej}, אז M' תפעל בצורה דומה. ■

הערה 4.6 אם המכונה המקורית מבצעת n צעדים, על צעד של המכונה המקורית אנו מבצעים $\mathcal{O}(n)$ צעדים, ובסה"כ $\mathcal{O}(n^2)$ צעדים. כלומר, עלות הסימולציה היא ריבועית - לא כל כך נורא.

הגדרה 4.7 מכונת טיורינג עם סרט דו־מימדי היא מכונה שהסרט שלה גדל בשני כיוונים - למטה וימינה. (אפשר להסתכל עליה בתור מטריצה "אינסופית".) מילת הקלט מגיעה על השורה הראשונה, הראש הקורא/כותב מתחיל בתא $(1, 1)$ ופונקציית המעברים היא $\delta : Q \times \Gamma \to Q \times \Gamma \times \{U, D, L, R\}$.
נדרוש גם שלא "ניפול" - אם הגענו לעמודה הראשונה והולכים שמאלה, אפשר לא לזוז. כנ"ל לגבי עלייה למעלה כאשר הגענו לשורה הראשונה.

טענה 4.8 למכונת טיורינג עם סרט דו־מימדי יש מכונת טיורינג שקולה עם סרט אחד.

הוכחה: אם יש מכונת טיורינג עם סרט אינסופי בשני הכיוונים, קל לבנות ממנה מכונה עם סרט דו־מימדי בעלת אותה השפה. פשוט צריך "לקפל" את הסרט סביב נקודת ההתחלה.
להיפך, בהינתן מכונת טיורינג עם סרט דו־מימדי יש לנו את הפונקציה f (החשיבה - ר' להלן) שיכולה להתאים כל תא מהסרט הדו־מימדי לתא אחד ויחיד על הסרט החד־מימדי. בתא הזה נשים את האות שהייתה בסרט הדו־מימדי באותו המקום. בנוסף, עלינו לקודד את המידע על כך שהראש נמצא במקום האמור בטבלה - ואפשר לצורך כך להחזיק עוד סרט אחד, למשל.
כדי לעשות את הסימולציה, נרוץ ונחפש היכן נמצא הראש, ואז נרוץ על הסרט הראשון עד האות שעליה הראש נמצא. כדי לרוץ ימינה/שמאלה/למעלה/למטה עלינו לחשב את האינדקס שעליו עמדנו (בשביל זה צריך לדעת לספור) ואז לבצע את החישוב הדרוש על האינדקס - למצוא את f^{-1} שלו, להוסיף ל־ i או ל־ j בהתאם לכיוון, לחשב את f של התוצאה ולפעול בהתאם. ■

וריאציות נוספות:
1. מכונת טיורינג עם סרט חד־כיווני - כלומר, הסרט חסום משמאל ולא חסום מימין. המודל הזה עדיין שקול למכונת טיורינג רגילה (תרגיל 5 שאלה 4). למשל, אפשר להשתמש בשני סרטים כדי לסמלץ את החלקים ה"שמאלי" וה"ימני" של הסרט; אפשר גם

לכתוב על הסרט החד־כיווני את $u\#v$ כאשר u החלק ה"שמאלי" ו־ v החלק הימני, ובכל פעם שהמכונה צריכה עוד מקום משמאל, מזיזים את המחרוזת ימינה.

2. אוטומט סופי דטרמיניסטי עם שתי מחסניות. המודל הזה שקול למכונת טיורינג רגילה (תרגיל 6 שאלה 1). למשל, אפשר לסמלץ אוטומט בעל שתי מחסניות בעזרת מכונת טיורינג בעלת שני סרטים (סרט אחד לשתי המחסניות, מופרדות ב־ $\#$ וסרט אחד לקלט); ואפשר לסמלץ מכונת טיורינג בעלת סרט דו־כיווני באמצעות אוטומט בעל שתי מחסניות, כך שמחסנית אחת מכילה את כל החלק של הסרט שמשמאל לראש והמחסנית השניה את כל החלק של הסרט שמימין לראש.

3. מכונת טיורינג עם סרט בגודל קבוע (LBA). המודל הזה אינו שקול (חלש יותר!) למכונת טיורינג רגילה. למעשה, LBA־ים מזהים בדיוק את כל השפות הרקורסיביות. (לא הוכחנו זאת פורמלית, אבל בהמשך ראינו ש־ A_{LBA} היא PSPACE־שלמה ובפרט כריעה, בעוד ש־ A_{TM} אינה כריעה.)

4.3 מכונת טיורינג אוניברסלית

זוהי מכונת טיורינג המקבלת כקלט מכונת טיורינג אחרת ומריצה אותה. ראשית צריך יהיה לקודד מכונת טיורינג כקלט. האי"ב יהיה $\Sigma_U = \{0, 1, \#\}$. נקודד את M באמצעות Σ_U:

- שמות המצבים יהיו מספרים טבעיים $Q = \{q_0, q_1, \ldots, q_n\}$ ונקודד אותם פשוט בתור מספרים, כאשר $\#$ מפרידה בין המצבים ו־ $\#\#\#$ מסמנת את סוף הקידוד:

$$\langle Q \rangle = 0000 \cdots 000 \# 0000 \cdots 001 \# 0000 \cdots 010 \# \cdots \# \# \#$$

- גם את אותיות Σ ו־ Γ נציג כמספרים טבעיים, שהרי גם $\Gamma \subseteq \Sigma$. נכתוב אותם בזה אחר זה, קודם $\langle \Sigma \rangle$ ואח"כ $\langle \Gamma \setminus \Sigma \rangle$ בקידוד בינארי, מופרדים ע"י $\#\#\#$.
- נקודד את δ בתור זוגות כאשר מספר האפשרויות ל־ $Q \times \Gamma$ הרי סופי. למשל, כדי לקודד את העובדה ש־ $\delta(q_1, \gamma_2) = (q_0, \sigma_1, L)$ נכתוב על הסרט לפי הסדר את $\langle q_1 \rangle, \langle \gamma_2 \rangle$ וכו':

$$0000 \cdots 001 \# 10 \# 0000 \cdots 000 \# 01 \# 0 \# \#$$

- לבסוף, כדי לקודד את q_0, q_{acc}, q_{rej} נכתוב אותם בזה אחר זה על הסרט.

מטרתנו: $M_U(\langle M \rangle, \langle w \rangle) = M(w)$. יהיו לנו שלושה סרטים - וניתן להראות בקלות שזה שקול למכונת טיורינג רגילה עם סרט אחד:

1. סרט הקלט של המכונה M_U שעליו יהיו כתובות $\langle M \rangle, \langle w \rangle$
2. סרט שיכיל את הסימולציה של הסרט של M
3. סרט שיכיל את המצב שבו M נמצאת

האלגוריתם:

1. שלב האתחול - נרצה לכתוב על סרט 2 את המילה $\langle w \rangle$ ועל סרט 3 את $\langle q_0 \rangle$. כדי לעשות זאת נרוץ ימינה בסרט 1 עד שספרנו 7 פעמים $\#\#\#$ ושם תהיה $\langle w \rangle$. מעתיקים אותה אות אחר אות, ומחזירים את הראש בסרט 1 למקום שבו מקודד $\langle q_0 \rangle$ ומעתיקים אותו לסרט 3.
2. שלב הסימולציה - ראשית בודקים האם הגענו ל־ q_{acc}, q_{rej} ע"י השוואת סרט 3 למה שכתוב בסרט 1 במקומות המתאימים. אחרת, צריך סימולציה של צעד δ, ומוצאים "מה צריך לעשות" בסרט 1, מעתיקים לסרטים 2,3 את התוצאה ומזיזים את הראש על סרט 2 ע"פ התוצאה.

נשים לב שניתן להרחיב את הרעיון הזה עוד - למשל, לבנות מכונה שמקבלת את השפה המשלימה לזו של M, או מכונה שמקבלת בנוסף מספר t שהוא מקסימום הצעדים שמותר לסימולציה לעשות, או חסם על גודל הסרט שאפשר להשתמש בו, וכו'.

4.4 מכונת טיורינג לא דטרמיניסטית

הגדרה 4.9 מכונת טיורינג לא דטרמיניסטית מוגדרת בדיוק כמו מכונת טיורינג, למעט העובדה ש־ δ היא פונקציה $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$.

הערה 4.10 משמעות הדבר שלמכונה כזו יכולות להיות כמה ריצות חוקיות על הקלט, ובכל שלב ייתכנו כמה מעברים חוקיים.

הגדרה 4.11 נאמר שמכונת טיורינג לא דטרמיניסטית מקבלת את המילה w אם קיימת ריצה מקבלת שלה על w. נאמר שהמכונה דוחה את w אם כל הריצות שלה על w דוחות. לבסוף, נאמר שהמכונה אינה עוצרת על w אם לא קיימת ריצה מקבלת, אך קיימת ריצה שאינה עוצרת.

הערה 4.12 כלומר, גם במקרה שהמכונה מקבלת את המילה, ייתכנו ריצות שאינן עוצרות.

טענה 4.13 תהי M מכונת טיורינג לא דטרמיניסטית. אזי קיימת מכונת טיורינג דטרמיניסטית D כך ש־ $L(D) = L(N)$.

הוכחה: אפשר להציג את הריצה של N על מילת קלט מסוימת בתור עץ שעוברים בו בין הקונפיגורציות שלה. כמובן, ייתכן שהעץ אינסופי.

נבנה את המכונה D בעזרת שלושה סרטים - סרט קלט, הסרט של המכונה N, וסרט הכתובת. לכל קונפיגורציה בעץ ניתן כתובת - ההחלטות שצריך לקבל כדי להגיע לקונפיגורציה הזאת מהקונפיגורציה ההתחלתית. (הכוונה ב"החלטה" לבחירה בין מספר אפשרויות במצב שבו למכונה יש מספר אפשרויות.)

בהינתן כתובת על סרט הכתובת אנחנו נדע לסמלץ על הסרט של N את המסלול המתאים לכתובת (שהוא דטרמיניסטי). כל שנותר הוא לעבור על הכתובות לפי סדר BFS, וברור שאפשר לעשות את זה. כאשר נתקלים בקונפיגורציה מקבלת - מקבלים. אם כל הענפים הסתיימו בקונפיגורציות דוחות - דוחים. ■

הערה 4.14 הקושי עם הבנייה הנ"ל הוא מבחינת זמן הריצה. בכל צומת יכול להיות מספר סופי של התפצלויות, לכל היותר $b = |Q \times \Gamma \times \{L, R\}|$. נתבונן למשל במקרה שבו המכונה מקבלת. נניח שהריצה המקבלת הקצרה ביותר על w היא באורך $g(|w|)$, אז D עלולה לעבור על $b^{\mathcal{O}(g(|w|))}$ קונפיגורציות, כלומר מספר אקספוננציאלי של פעולות ביחס למכונה המקורית.

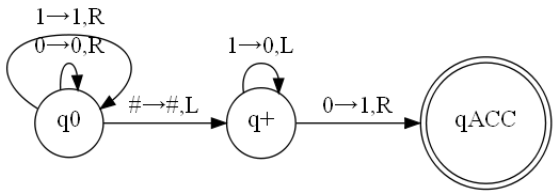
4.5 פונקציות חשיבות

במקום מכונת טיורינג שהפלט שלה הוא כן/לא, נסתכל על פונקציות.

הגדרה 4.15 פונקציה $f : \Sigma^* \to \Sigma^*$ היא חשיבה (computable) אם קיימת מכונת טיורינג M שמקבלת $w \in \Sigma^*$, רצה ותמיד עוצרת, והתוכן של הסרט הוא $f(w)$. (אפשרות שקולה היא שלמכונה יהיו שני סרטים.)

דוגמאות לפונקציות חשיבות:

1. למשל, ראינו (פחות או יותר) שפונקציה שמוסיפה 1 למספר הנתון בבינארי היא פונקציה חשיבה:



2. כמובן, הפונקציה לא חייבת לפעול על מספרים. למשל, פונקציה שמקבלת $\langle A \rangle$ קידוד של NFA ומחשבת $\langle B \rangle$ קידוד של DFA השקול לו היא פונקציה חשיבה - צריך לשתכנע שאלגוריתם הדטרמיניזציה אכן ניתן למימוש בעזרת מכונת טיורינג.

3. ראינו ש־ $|\mathbb{N}| = \aleph_0$ ובדיסקרטית מוכיחים ש־ $|\mathbb{N} \times \mathbb{N}| = \aleph_0$ ע"י כך שיוצרים פונקציה חח"ע ועל מ־ $\mathbb{N}$ ל־ $\mathbb{N} \times \mathbb{N}$. נרצה עכשיו להסתכל על פונקציה אחת כזאת:

i/j	1	2	3	4	···
1	1	3	6	10	
2	2	5	9		
3	4	8			
4	7				
⋮					

אפשר לכתוב לה גם נוסחא סגורה:

$$f(i,j) = \frac{(i+j-2)(i+j-1)}{2} + j$$

הפונקציה הזאת, כמובן, חשיבה - אנו יודעים לחסר, לחבר, לכפול ולחלק בשתיים (שלמים) באמצעות מכונת טיורינג.

4. לפונקציה (3) יש גם הפכית - אפשר לכתוב לה במפורש נוסחא אריתמטית והיא תהיה חשיבה. אולם אנו נרצה אלגוריתם אחר - נניח שרוצים למצוא את $f^{-1}(k)$:

- נכתוב על הסרט את המשבצת השמאלית עליונה, נפעיל את f ונקבל $f(1,1)$.
- אם קיבלנו את k - דיינו.
- אחרת, נתקדם לתא הבא - $(2,1)$ ומשם ל־ $(1,2)$ ומשם ל־ $(3,1)$ וכן הלאה.

האופן המדויק שבו מקדמים את (i,j) הוא כזה - אם $i = 1$ אז מוסיפים לו 1 ואת j מחזירים ל־ 1; אחרת, מקדמים את j ומחסרים 1 מ־ i.

כיוון ש־ k מספר טבעי, מובטח שהמכונה תעצור בסופו של דבר. לכן f^{-1} חשיבה.

5. נבנה פונקציה $f : \{\langle M \rangle\} \to \{\langle M \rangle\}$ המקבלת קידוד של מכונת טיורינג ומייצרת קידוד של מכונת טיורינג בעלת אותה שפה שלעולם לא מגיעה למצב הדוחה. כיצד הפונקציה פועלת? כאשר המכונה המקורית פונה ל־ q_{rej} שלה, המכונה החדשה תפנה למצב שבו היא תיתקע בלולאה אינסופית.

4.6 Random Access Machine

נרצה להראות שקילות בין מכונות טיורינג, שעדיין לא נראות כל כך כמו מחשב, לבין מודל שלכאורה נראה חזק יותר ומאוד דומה למחשב, אבל למעשה לא מציע שום כוח חישוב נוסף.

- ל־ RAM יש מערך זיכרון אינסופי וממוספר, כאשר התוכן של כל תא הוא מספר טבעי.
- התוכן ההתחלתי בכל תא הוא 0.
- יש פונקציה $c : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$ שמחזירה את התוכן של כל תא.
- יש תוכנית שהמכונה מריצה, ששמורה מחוץ לזיכרון.
- יש קלט - סדרה סופית של מספרים טבעיים.
- יש מונה (PC) שזוכר איפה בריצת התוכנית אנחנו נמצאים.
- יש פונקציה v שאומרת מה הערך של האופרנד בפקודות שמיד נראה. הפונקציה מוגדרת כך:
  - $v(\text{"=x"}) = x$ - אופרנד ששווה למספר קבוע כלשהו
  - $v(\text{"x"}) = c(x)$ - אופרנד שקורא זיכרון ממקום כלשהו
  - $v(\text{"*x"}) = c(c(x))$ - אופרנד שמבצע מיעון עקיף מהזיכרון

הפקודות האפשריות של המכונה הן:

הפקודה	המשמעות
LOAD(op)	$c(0) \leftarrow v(op)$
STORE(op)	$c(v(op)) \leftarrow c(0)$
ADD(op)	$c(0) \leftarrow c(0) + v(op)$
SUB(op)	$c(0) \leftarrow c(0) - v(op)$
READTO(op)	$c(v(op)) \leftarrow$ NextInput
JUMP(op)	$PC \leftarrow v(op)$
JUMPIF > 0(op)	If $c(0) > 0$ then $PC \leftarrow v(op)$
ACCEPT	Accept
REJECT	Reject

טענה 4.16 קיימת מכונת טיורינג שמקבלת קידוד של תוכנית ל־ RAM וקלט ואומרת האם התוכנית מקבלת או דוחה את הקלט.

הוכחה: (סקיצה) נצטרך 9 סימנים לסימון סוגי הפקודות, קידוד של האופרנדים, וכן קידוד של הקלט. את כל אלה נרשום ונפריד ביניהם עם $\#$ כרגיל - על סרט הקלט. על סרט נוסף נחזיק ייצוג של הזיכרון של המכונה שמריצה את התוכנית, ועוד סרטים לפי הצורך עבור חישובים. על הסרט הראשון נסמן גם על איזו פקודה אנחנו נמצאים. על סרט הזיכרון נרשום זוגות (i,j) עם המשמעות שבתא ה־ i בזיכרון נמצא הערך j, ובתנאים שלא נשתמשנו בהם נניח שמופיע 0.

כעת עלינו ללמוד לבצע את הפקודות של התוכנית.

1. טיפול באופרנד - נרצה לשים על סרט החישוב השלישי את $v(op)$. עבור "$= x$" פשוט נעתיק את המספר x לסרט החישוב. במקרים שדרושה גישה לזיכרון, נצטרך לרוץ על סרט הזיכרון ולחפש את הזוג (x,j) ולהעתיק את j לסרט החישוב (או 0 אם הוא לא נמצא). עבור מיעון עקיף, נצטרך לחזור על זה פעמיים.
2. טיפול בפקודות - עבור פקודות הקריאה/כתיבה מהזיכרון, מחשבים את $v(op)$ ומחפשים אותו בסרט הזיכרון. את הפעולות האריתמטיות אנחנו במילא יודעים לעשות. עבור הפעולה READTO נצטרך לרוץ על w ולזכור מה עוד לא קראנו מתוכה, למשל ע"י מחיקת החלקים שכבר קראנו. עבור הפעולה JUMP נצטרך לעבור על הסרט הראשון למקום המתאים בתוכנית, וכנ"ל עבור JUMPIF > 0.

לגבי ביצועים - על סרט הזיכרון, אם המכונה עשתה n צעדים, יש n תאים לכל היותר שכל אחד מכיל מספר עד k, ואנו משתמשים ב־ $\mathcal{O}(n \log k)$ מקום. מספר מעברי ה־ δ יהיה $\mathcal{O}(n^2 \log k)$ כי יש n צעדים. בשאר הסרטים המקום די קבוע. כלומר, הסימולציה שלנו היא במספר פולינומיאלי של צעדים ביחס למספר הצעדים של המכונה המקורית. ■

5 כריעות ואי־כריעות

לאחר שראינו את המודל של מכונת טיורינג, נשאלת השאלה האם מודל זה יכול להכריע את כל השפות. אפילו משיקולי ספירה ניתן לראות שאין זה המצב; בפרק זה נגיע למסקנות פורמליות בנושא.

5.1 המחלקות R, RE ואי־כריעות

הגדרה 5.1 נסמן ב־ R (שפות רקורסיביות, recursive) את מחלקת השפות הניתנות להכרעה ע"י מכונת טיורינג. נסמן ב־ RE (שפות הניתנות למניה רקורסיבית, recursively enumerable) את מחלקת השפות שניתנות לזיהוי ע"י מכונת טיורינג.

נסמן ב־ $coRE$ את מחלקת השפות L כך ש־ $\overline{L} \in RE$. (במילים אחרות, קיימת מכונת טיורינג שעוצרת ומקבלת מילים שאינן ב־ L, אבל על מילים ב־ L היא דוחה או נתקעת.)

הערה 5.2 כמובן, $R \subseteq RE$. בהמשך נראה ש־ $RE \setminus R \neq \emptyset$.

משפט 5.3 $RE \cap coRE = R$ (כלומר, אם אפשר לזהות שפה ואת המשלים שלה - אזי היא ניתנת להכרעה).

הוכחה: תחילה נראה ש־ $R \subseteq RE \cap coRE$. אנו יודעים שאם $L \in R$ אז גם $\overline{L} \in R$, שהרי אם במכונת טיורינג נחליף את q_{acc} ב־ q_{rej} ולהיפך נקבל מ"ט עבור השפה המשלימה. מכאן המסקנה.

כעת נראה ש־ $RE \cap coRE \subseteq R$. אם M מזהה את L ו־ $\overline{M}$ מזהה את $\overline{L}$, נרצה להריץ במקביל את שתי המכונות ומובטח שאחת מהן תעצור על כל מילה. נעשה זאת כך:

המכונה החדשה M', על המילה w:

1. תאתחל $i \leftarrow 1$
2. הרץ את M על w במשך i צעדים: אם היא קיבלה, עצור וקבל
3. הרץ את $\overline{M}$ על w במשך i צעדים: אם היא קיבלה, עצור ודחה
4. $i \leftarrow i+1$ וחזור לשלב 2

ברור שכל איטרציה של הלולאה באמת עוצרת בזמן סופי, וכן ברור שנעצור בסופו של דבר ונצא מהלולאה - הרי אם $w \in L$ אז קיים i כך ש־ M עוצרת ומקבלת את w תוך i צעדים, ואם $w \notin L$ אז כנ"ל לגבי $\overline{M}$. לכן M' מכריעה את L, כלומר $L \in R$. ■

משפט 5.4 קיימת שפה $L \notin R$.

הוכחה: משיקולי ספירה. למשל, קבוצת כל השפות מעל $\{0,1\}$ היא מעוצמה $2^{\aleph_0}$. לעומת זאת, קבוצת כל מכונות הטיורינג מעל אותה שפה היא קבוצה בת מניה, שכן מכונת טיורינג ניתנת לתיאור (קידוד) סופי. מכאן, יש שפה שאין מכונת טיורינג שמכריעה אותה. (זה גם אומר שיש שפה $L \notin RE$.) ■

הערה 5.5 אפשר להשתמש באותו טיעון גם להרבה דברים מעניינים נוספים. למשל, יש מספר ממשי שלא ניתן לחשב, כלומר שלא קיימת תוכנית שמדפיסה אותו. זה שוב משיקולי ספירה - תוכנית מחשב ניתנת לקידוד סופי, ולכן קבוצת תוכניות המחשב היא בת מניה, ואילו קבוצת המספרים הממשיים אינה בת מניה.

משפט 5.6 $R \subsetneq RE$.

הוכחה: נצביע במפורש על השפה $A_{TM} = \{\langle M, w\rangle : \ M \text{ accepts } w\}$ ונראה ש־ $A_{TM} \in RE$ אבל $A_{TM} \notin R$. תחילה, $A_{TM} \in RE$ שכן קיימת M_{TM} מכונת טיורינג שמזהה את A_{TM} ופועלת כך: היא מריצה את M על w, ופועלת כמוה. אכן, M_{TM} תקבל את $\langle M, w\rangle$ אסם M מקבלת את w.

כעת נניח בשלילה שקיימת מכונת טיורינג H שמכריעה את A_{TM}, כלומר על קלט $\langle M, w\rangle$ היא מקבלת אסם M מקבלת את w, ואחרת - דוחה (אבל תמיד עוצרת). נבנה בעזרתה את המכונה D שמקבלת קידוד $\langle M\rangle$ של מכונת טיורינג ופועלת כך:

$$D(\langle M\rangle) = \begin{cases} \mathcal{ACC} & M(\langle M\rangle) = \mathcal{ACC} \\ \mathcal{REJ} & M(\langle M\rangle) = \mathcal{REJ} \end{cases}$$

אופן הפעולה של D הוא כזה: על קלט $\langle M\rangle$, היא כותבת על הסרט $\langle M\rangle\langle M\rangle$ ואז מריצה את H על הקלט החדש. לבסוף, נבנה מ־ D את המכונה D' הפועלת כך:

$$D'(\langle M\rangle) = \begin{cases} \mathcal{REJ} & M(\langle M\rangle) = \mathcal{ACC} \\ \mathcal{ACC} & M(\langle M\rangle) = \mathcal{REJ} \end{cases}$$

את זה אנחנו כבר יודעים לעשות ע"י החלפת המצב המקבל והדוחה ב־ D. כעת נותר רק להריץ את D' על עצמה:

$$D'(\langle D'\rangle) = \begin{cases} \mathcal{REJ} & D'(\langle D'\rangle) = \mathcal{ACC} \\ \mathcal{ACC} & D'(\langle D'\rangle) = \mathcal{REJ} \end{cases}$$

שתי האפשרויות לא ייתכנו, ולכן קיבלנו סתירה. לכן, $A_{TM} \notin R$. ■

מסקנה 5.7 $A_{TM} \notin coRE$.

המסקנה היא מרחיקת לכת. יש בעיות שלא ניתן להכריע אותן - ויש לנו כעת הכלי התיאורטי כדי להראות את זה. אז הנה דוגמה נוספת, שנשתמש בה ברדוקציה (נגדיר בהמשך במדויק מהי רדוקציה).

טענה 5.8 $HALT_{TM} = \{\langle M, w\rangle : \ M \text{ halts on } w\} \notin R$ (בעיית העצירה).

הוכחה: נראה שאם $HALT_{TM} \in R$ אז גם $A_{TM} \in R$ ונקבל את הסתירה. ובכן אם $HALT_{TM} \in R$ אז תהי T מ"ט שמכריעה את $HALT_{TM}$. נבנה מ"ט S שמכריעה את A_{TM} כך: על קלט $\langle M\rangle, w$:

1. נריץ את T על הזוג $\langle M\rangle, w$. אם T דוחה, נדחה.
2. אם T מקבלת את $\langle M\rangle, w$ אז נריץ את M על w ונענה כמוה (מובטח לנו ש־ M תעצור!). ■

טענה 5.9 $REG_{TM} = \{\langle M\rangle : \ M \text{ is a TM}, \ L(M) \in REG\} \notin R$.

הוכחה: נניח בשלילה שיש מ"ט T המכריעה את REG_{TM} ונבנה מ"ט S שמכריעה את A_{TM}. בהינתן $\langle M\rangle, w$ נייצר מ"ט M' מעל הא"ב $\{0,1\}$ אשר על קלט $x \in \{0,1\}^*$ פועלת כך:

1. אם x מהצורה 0^n1^n, נקבל.
2. אחרת, נריץ את M על w ונקבל אסם M מקבלת את w.

כעת נשים לב שאם M לא מקבלת את w, אז $L(M') = \{0^n1^n : \ n \geq 0\} \notin REG$. לעומת זאת, אם M מקבלת את w, אז $L(M') = \Sigma^*_{M'} \in REG$. כלומר, $M' \in REG_{TM} \iff L(M') \in REG \iff (\langle M\rangle, w) \in A_{TM}$ ובזאת השלמנו את הבנייה. ■

5.2 סגירות R ו־ RE

טענה 5.10

1. $L, M \in RE \implies L \cup M \in RE$.
2. $L, M \in RE \implies L \cap M \in RE$.
3. $L \in R \implies \overline{L} \in R$.
4. $L \in RE \implies L^* \in RE$.
5. $L, M \in RE \implies L \cdot M \in RE$

הוכחה: 1. מריצים את המכונות "במקביל" ומקבלים אם אחת מהן מקבלת. (ההרצה "במקביל" משמעותה שמבצעים t צעדים של כל אחת מהמכונות, מסתכלים על התוצאה, ואז מקדמים את t במידת הצורך.)
2. מריצים את שתי המכונות "במקביל" ומקבלים אם שתיהן מקבלות. (כאן אין אפילו צורך בהרצה אינקרמנטלית, כי אם אחת המכונות נתקעת, במילא המילה אינה בחיתוך השפות.)
3. מחליפים בין q_{acc} ל־ q_{rej}. נשים לב שזה לא נכון לגבי RE (למשל, $A_{TM} \in RE$ אבל $\overline{A_{TM}} \notin RE$, שכן אחרת $A_{TM} \in coRE$ ואז $A_{TM} \in RE \cap coRE = R$).
4. נשתמש במכונת טיורינג עם סרט דו־מימדי. כשמקבלים את מילת הקלט w, נכתוב את כל החלוקות האפשריות שלה ל־ $w_1, \ldots, w_k$, כל חלוקה על שורה משלה של הסרט. עבור t הולך וגדל, נריץ את המכונה של L במקביל על כל שורה (כאשר בכל שורה, L רצה במקביל על כל החלקים) במשך t צעדים. אם $w \in L^*$, יהיה t ושורה מתאימה שעבורם נקבל. לכן, $L^* \in RE$.
אפשרות אחרת היא להשתמש במכונת טיורינג לא דטרמיניסטית, שבוחרת בצורה לא דטרמיניסטית חלוקה של w ומריצה עליה את L כנ"ל. (ובכל מקרה, ר' תרגיל 6 שאלה 4.)
אם מתמקדים ב־ R, אפשר גם להשתמש באלגוריתם תכנון דינאמי (ר' תרגיל 9, שאלה 5). הרעיון הוא להסתכל על כל התת־מחרוזות ולבדוק האם הן ב־ L^*, כאשר מקרה הבסיס הוא התת־מחרוזת הריקה ותת־מחרוזות שהן ב־ L. זה מאפשר גם לקבל זמן ריצה יעיל - ריבועי בזמן הריצה של המכונה המכריעה את L.
5. בדומה ל־ L^*, בהינתן מילה w נסתכל על כל המקומות i שבהם ניתן לחלק את המילה לשני חלקים, ונכתוב כל אפשרות על שורה משלה. עבור t הולך וגדל, נריץ את המכונה של L והמכונה של M במקביל על כל שורה (כאשר בכל שורה, L רצה על החלק הראשון ו־ M על החלק השני) במשך t צעדים. ■

5.3 רדוקציית מיפוי

הגדרה 5.11 שפה $A \subseteq \Sigma^*$ ניתנת לרדוקציית מיפוי לשפה $B \subseteq \Sigma^*$ ונסמן $A \leq_m B$ אם קיימת פונקציה חשיבה $f : \Sigma^* \to \Sigma^*$ כך שלכל $w \in \Sigma^*$ מתקיים $w \in A \iff f(w) \in B$.

הערה 5.12 רדוקציית מיפוי אינה סימטרית, כלומר אם $A \leq_m B$ אין הכרח שגם $B \leq_m A$. למשל, $01^* \leq_m A_{TM}$ אבל ממש לא ההיפך. זאת משום שההפכית של פונקציה חשיבה אינה בהכרח חשיבה/קיימת!

משפט 5.13 אם $A \leq_m B$ ו־ $B \in R$, אז $A \in R$.

הוכחה: בהינתן מכונת טיורינג M_B המכריעה את B נבנה מ"ט M_A המכריעה את A. היא תפעל כך על הקלט w: תחשב את $f(w)$ ותריץ את M_B על התוצאה. M_A תחזיר את את התשובה של M_B. ■

מסקנה 5.14 אם $A \leq_m B$ ו־ $A \notin R$, אז $B \notin R$.

טענה 5.15 אם $A \leq_m B$ ו־ $B \in RE$, אז $A \in RE$.

הוכחה: באופן סימטרי לחלוטין ל"משפט" שראינו קודם. ■

מסקנה 5.16 1. אם $A \leq_m B$ ו־ $A \notin RE$, אז $B \notin RE$.
2. אם $A \leq_m B$ ו־ $B \in coRE$, אז $A \in coRE$.
3. אם $A \leq_m B$ ו־ $A \notin coRE$, אז $B \notin coRE$.

טענה 5.17 (טריוויאלית) רדוקציית מיפוי היא טרנזיטיבית, כלומר $A \leq_m B \wedge B \leq_m C \implies A \leq_m C$.

<u>דוגמאות לרדוקציות מיפוי פשוטות</u>:
1. פורמליזציה של $A_{TM} \leq_m REG_{TM}$. בנינו את הפונקציה החשיבה שמייצרת מ־ $(\langle M \rangle, w)$ את המ"ט M', ומתקיים $(\langle M \rangle, w) \in A_{TM} \iff f(\langle M \rangle, w) \in REG_{TM} \iff M' \in REG_{TM} \iff L(M') \in REG$.
2. פורמליזציה של $A_{TM} \leq_m HALT_{TM}$. נבנה את הפונקציה החשיבה שמייצרת מ־ $(\langle M \rangle, w)$ את המ"ט M' שמריצה את M על w, מקבלת אם M מקבלת את w ואחרת נכנסת ללולאה אינסופית (כלומר אפילו אם M דוחה, M' נכנסת ללולאה אינסופית). אז מתקיים $(\langle M \rangle, w) \in A_{TM} \iff M' \in HALT_{TM}$.
3. נראה ש־ $A^{\varepsilon}_{TM} = \{\langle M \rangle : M \text{ accepts } \varepsilon\}$ אינה כריעה באמצעות רדוקציה $A_{TM} \leq_m A^{\varepsilon}_{TM}$. בהינתן זוג $\langle M \rangle, w$ הרדוקציה תפלוט מכונה $\langle M' \rangle$ שתפעל כך: בתחילת הריצה היא תדפיס את המילה w על הסרט, תחזיר את הראש לתחילת הסרט, ותמשיך לרוץ על פי כללי המעבר של M. קל לראות ש־ $(\langle M \rangle, w) \in A_{TM} \iff \langle M' \rangle \in A^{\varepsilon}_{TM}$.

טענה 5.18 $INF_{TM} = \{\langle M \rangle : L(M) \text{ is infinite}\} \notin RE \cup coRE$.

הוכחה: תחילה נראה ש־ $A_{TM} \leq_m INF_{TM}$. ובכן, עבור $\langle M\rangle, w$ נבנה מ"ט T שפועלת כך על קלט x:
1. T מסמלצת את M על w.
2. אם M דוחה את w, T דוחה;
3. אחרת, אם $|x|$ זוגי אז T מקבלת[4], ואחרת T דוחה.

מקבלים ש־ $L(T) = \emptyset$ אם M לא מקבלת את w, ואחרת $L(T)$ אינסופית. כלומר, $(\langle M\rangle, w) \in A_{TM} \iff \langle T\rangle \in INF_{TM}$.
לכן $INF_{TM} \notin coRE$, שהרי $A_{TM} \notin coRE$.

כעת נראה ש־ $\overline{A_{TM}} \leq_m INF_{TM}$. בהינתן $\langle M\rangle, w$ נבנה מ"ט T שפועלת כך על קלט x:
1. אם M מקבלת את w תוך $|x|$ צעדים, אז T דוחה.
2. אחרת, T מקבלת.

כעת אם M לא מקבלת את w, הרי ש־ $L(T) = \Sigma^*$ ובפרט אינסופית. לעומת זאת, אם M מקבלת את w אז יש ריצה באורך t המקבלת את w, ולכן $L(T) \subseteq \{x : |x| < t\}$, ובפרט סופית. קיבלנו ש־ $\langle T\rangle \in INF_{TM} \iff (\langle M\rangle, w) \in \overline{A_{TM}}$. לכן $INF_{TM} \notin RE$, שהרי $\overline{A_{TM}} \notin RE$. ■

טענה 5.19 A_{TM} היא RE־שלמה, כלומר לכל $L \in RE$ מתקיים $L \leq_m A_{TM}$.

הוכחה: תרגיל 8, שאלה 4. (זה ממש קל.) ■

טענה 5.20 $ALL_{TM} = \{\langle M\rangle : \ L(M) = \Sigma^*\} \notin RE \cup coRE$

מסקנה 5.21 $EQ_{TM} = \{\langle M_1, M_2\rangle : \ L(M_1) = L(M_2)\} \notin RE \cup coRE$

הוכחה: (תרגיל 8, שאלה 3) מדובר ברדוקציות הבסיסיות ביותר מ־ A_{TM} ו־ $\overline{A_{TM}}$. למשל, $A_{TM} \leq_m ALL_{TM}$ על ידי כך שמ־ $\langle M, w\rangle$ בונים מכונה M' שעל כל קלט מריצה את M על w ועונה כמוה. ולהיפך, $\overline{A_{TM}} \leq ALL_{TM}$ על ידי כך שמ־ $\langle M, w\rangle$ בונים מכונה M' שעל קלט $|x|$ מריצה את M על w במשך $|x|$ צעדים, ומקבלת אסם M לא מקבלת. את המסקנה אפשר להוכיח ישירות, או ע"י רדוקציה $ALL_{TM} \leq_m EQ_{TM}$ באמצעות מכונה שמקבלת את Σ^*. ■

5.4 אנומרטורים (מונים)

הגדרה 5.22 <u>אנומרטור</u> (enumerator) זו מכונת טיורינג עם מדפסת. אופי העבודה - יש למכונה סרט עבודה אבל מדי פעם היא כותבת משהו על סרט הפלט שלה, שממנו היא לא יכולה לקרוא או למחוק. מילה היא בשפה של האנומרטור אסם היא מודפסת בסופו של דבר ע"י האנומרטור.

משפט 5.23 $L \in RE$ אסם קיים אנומרטור E כך ש־ $L(E) = L$.

הוכחה: ($\Rightarrow$) בהינתן E נייצר M שמזהה את $L(E)$. היא תפעל כך: בהינתן מילה w, M מריצה את E, ובכל פעם ש־ E מדפיסה מילה y, M בודקת האם $w = y$ ואם כן - עוצרת ומקבלת.

($\Leftarrow$) בהינתן M שמזהה את L נבנה E כך ש־ $L(E) = L$. אנו יודעים ש־ Σ^* בת מניה, ונתבונן בסידור (למשל, לקסיקוגראפי) של Σ^*. לסידור נקרא $w_0, w_1, \ldots$. כעת E תפעל כך: עבור $i = 1, 2, \ldots$ היא תריץ את M על המילים $w_0, w_1, \ldots, w_i$ במשך i צעדים על כל מילה. אם M קיבלה מילה מסוימת, E תדפיס אותה.

ואמנם, כל מילה w שמתקבלת ע"י M, מתקבלת תוך איזה j צעדים, ותודפס ע"י E אינסוף פעמים החל מהאיטרציה ה־ j. ■

טענה 5.24 תהי $L \in RE$ שפה אינסופית. אז קיימת שפה אינסופית $L' \subseteq L$ כך ש־ $L' \in R$. (מסיכום תרגול 7)

הוכחה: יהי E האנומרטור של L. נגדיר את השפה

$$L' = \{w \in L : \ E \text{ only prints words shorter than } w \text{ before printing } w\}$$

קל לראות שהשפה אינסופית, שכן אם היא הייתה סופית יש בה את המילה הארוכה ביותר w' ו־ E לא מדפיס מילים יותר ארוכות מ־ w' לאחר הדפסתה - בסתירה לכך ש־ L אינסופית.

כמו כן, $L' \in R$ - נבנה מ"ט שבהינתן מילה w מריצה את E כל עוד מודפסות מילים שקצרות מ־ w. יש מספר סופי של כאלה. אם מודפסת מילה ארוכה יותר, עוצרים ודוחים; אם w הודפסה, מקבלים. ■

[4] הבחירה בתנאי הזה שרירותית לחלוטין.

5.5 אוניברסאליות של דקדוקים חסרי הקשר

נרצה לבדוק האם ניתן להכריע באמצעות מכונת טיורינג את השפה:

$$ALL_{CFG} = \{\langle G\rangle : \ G \in CFG \wedge L(G) = \Sigma_G^*\}$$

כלומר, לזהות דקדוקים חסרי הקשר שניתן לגזור מהם את כל המילים.
ראינו קודם אלגוריתם לבדיקת שייכות של מילה לדקדוק, זה היה אלגוריתם תכנון דינאמי. מהאלגוריתם הנ"ל ניתן להסיק ש־ $ALL_{CFG} \in coRE$ באמצעות מכונת טיורינג כדלקמן:

- על כל מילה $w \in \Sigma_G^*$ (למשל, בסדר לקסיקוגראפי) נבדוק האם $w \in L(G)$. אם לא, נדחה את $\langle G\rangle$.
- אחרת, ממשיכים למילה הבאה.

ברור שאם $L(G) \neq \Sigma^*$, בסופו של דבר נגיע למילה $w \notin L(G)$ ונדחה ולכן $ALL_{CFG} \in coRE$.
אבל כעת נראה ש־ $ALL_{CFG} \notin R$ (למעשה מספיק להראות ש־ $ALL_{CFG} \notin RE$).

משפט 5.25 לכל מ"ט M מעל א"ב Σ_M וקלט $w \in \Sigma_M^*$ קיים A אוטומט מחסנית כך ש־ $L(A) = \Sigma_A^*$ אם M לא מקבלת את w.

הוכחה: נסתכל על הריצה של M על w, המורכבת מסדרת קונפיגורציות $c_0, c_1, \ldots, c_k$. כל קונפיגורציה כזו אפשר לקודד - היא מורכבת ממצב הסרט (מספר סופי של אי־$\sqcup$), מיקום הראש, והמצב באוטומט. נפריד בין הקונפיגורציות באמצעות $\#$ למשל. נדבר ספציפית של שפת הקידודים של סדרת קונפיגורציות שהיא תקינה עבור המכונה M והמילה w ומקבלת. סדרה כזאת קיימת אסם M מקבלת את w; אחרת, אין כזאת.
קידוד של סדרת קונפיגורציות תקינה ומקבלת מקיים:
1. c_0 היא הקונפיגורציה ההתחלתית של M על w. (המצב הוא q_0, הסרט מכיל את w, והראש נמצא בתחילת הסרט.)
2. c_k היא קונפיגורציה מקבלת. (המצב הוא q_{acc}.)
3. כל מעבר מ־ c_i ל־ c_{i+1} תקין ב־ M.

לכן קידוד שאינו כזה מקיים לפחות אחד משלושת הבאים:
1. c_0 אינה הקונפיגורציה ההתחלתית.
2. c_k אינה קונפיגורציה מקבלת.
3. קיים מעבר $c_i \to c_{i+1}$ שאינו תקין.

נרצה לבנות אוטומט מחסנית שמקבל בדיוק את סדרות הקונפיגורציות שמקיימות אחד משלושת התנאים הנ"ל. ניתן לעשות זאת לכל תנאי בנפרד כי אוטומט מחסנית הוא לא דטרמיניסטי ולכן יכול "להתפצל":
1. c_0 היא מחרוזת קבועה כלשהי, אז אפשר כמובן לבנות "ענף" של האוטומט שיבדוק האם הוא קיבל משהו שאינו המחרוזת הקבועה הזאת.
2. בכל פעם שרואים $\#$ (המפריד בין קונפיגורציות), "ננחש" באופן לא דטרמיניסטי שהגענו לקונפיגורציה האחרונה ובודקים האם המצב שלה מקבל. אם כן, צריך לבדוק שבאמת הגענו לקונפיגורציה האחרונה כדי להחליט.
3. נצטרך "לנחש" בכל פעם מהו ה־ i שעבורו בודקים את התקינות. חלק מהעניין זה לבדוק שמצב הסרט לא השתנה בין שתי קונפיגורציות, וזה קצת דומה לשפה $\{ww\}$ שהיא לא חסרת הקשר! כדי להתמודד עם זה, לכל קונפיגורציה אי־זוגית על הסרט נעשה reverse -

$$\langle c_0\rangle\#\langle c_1\rangle^R\#\langle c_2\rangle\#\langle c_3\rangle^R\#\ldots$$

עכשיו זה כבר יותר דומה לשפה $\{ww^R\}$ שהיא חסרת הקשר - חוץ ממקום אחד שבו הסרט השתנה.
לסיכום, ראינו סקיצה של PDA שמקבל בדיוק את סדרת הקונפיגורציות הלא תקינות ו/או לא מקבלות של המכונה M. הדבר היחיד שהאוטומט לא מקבל זו הריצה התקינה והמקבלת של M, אם יש כזו. לכן $L(A) = \Sigma_A^*$ אסם M לא מקבלת את w, כנדרש. ■

מסקנה 5.26 $ALL_{CFG} \notin R$.

הוכחה: נניח בשלילה שכך המצב, אז יש מ"ט M_0 שמכריעה את ALL_{CFG}. כעת נבנה מ"ט M_1 שמכריעה את A_{TM} ונגיע לסתירה (זוהי דוגמה לשימוש ברדוקציה באופן לא פורמלי). M_1 תבנה את ה־ PDA מהמשפט ותכתוב את הקידוד שלו על הסרט. לאחר מכן היא תהפוך את הקידוד הזה לקידוד של CFG (ראינו שזה אפשרי). נריץ את M_0 על הקידוד הזה ונדע האם ה־ PDA היה אוניברסאלי. אם M_0 מחזירה "כן", נחזיר "לא" ולהיפך. לפי המשפט, זה בדיוק "האם M מקבלת את w", כלומר הכרענו את A_{TM} וזו סתירה לכך ש־ $A_{TM} \notin R$. ■

5.6 משפט רייס

טענה 5.27 $L_1 = \{\langle M\rangle : \exists w \in L(M),\ |w| > 5\} \notin coRE$.

הוכחה: נראה רדוקציה $A_{TM} \leq_m L_1$. מהקלט $\langle M\rangle, w$ נייצר את המכונה T שפועלת על קלט x כך:
1. T מסמלצת את M על w.
2. אם M דוחה, T דוחה;
3. אחרת, אם $x = 0101101$ אז T מקבלת;
4. אחרת, T דוחה.

קיבלנו ש־ $L(T) = \{0101101\}$ כאשר M מקבלת את w, ו־ $L(T) = \emptyset$ אחרת. לכן זוהי רדוקציה, ומכאן המסקנה. ■

נראה מתבקש שתהיה הכללה כלשהי לטענות מהסוג הזה, שתאפשר לטעון משהו על שפות של מכונות טיורינג. לצורך כך נצטרך הגדרה:

הגדרה 5.28 P קבוצה של מכונות טיורינג נקראת <u>תכונה</u>. תכונה נקראת <u>סמנטית</u> אם היא תלויה בשפה של המכונה, כלומר לכל שתי מכונות טיורינג M_1, M_2 אם $L(M_1) = L(M_2)$ אזי $M_1 \in P \iff M_2 \in P$. נאמר שתכונה היא <u>טריוויאלית</u> אם $P = \emptyset$ או P היא קבוצת כל המכונות טיורינג.
נסמן ב־ T_0 מ"ט המקיימת $L(T_0) = \emptyset$. נסמן $L(P) = \{\langle M\rangle : M \in P\}$.

למה 5.29 תהי P תכונה סמנטית לא טריוויאלית. נניח ש־ $T_0 \notin P$. אזי $A_{TM} \leq_m L(P)$. (ולכן גם $L(P) \notin coRE$.)

הוכחה: תהי $H \in P$ מכונה כלשהי (יש כזו כי P אינה טריוויאלית). עבור $\langle M\rangle, w$ נבנה מכונת טיורינג T הפועלת על הקלט x כך:
1. T מסמלצת את M על w.
2. אם M דוחה, אז T דוחה;
3. אחרת, T מריצה את H על x ועונה כמוה.

התוצאה היא שאם M לא מקבלת את w, אז $L(T) = L(T_0) = \emptyset$ ולכן $T \notin P$ כי P היא תכונה סמנטית. לחלופין, אם M מקבלת את w אז $L(T) = L(H)$ ושוב $T \in P$ כי $H \in P$. בזאת השלמנו את הרדוקציה. ■

משפט 5.30 (משפט רייס) תהי P תכונה סמנטית לא טריוויאלית. אז השפה $L(P) = \{\langle M\rangle : M \in P\}$ אינה כריעה.

הוכחה: כדי להוכיח את המשפט עלינו לטפל רק במקרה שבו P מכילה גם מכונות בעלות שפה ריקה. ובכן, אם $T_0 \in P$ אז התכונה המשלימה $\overline{P}$ מקיימת $T_0 \notin \overline{P}$ ולפי הלמה $L(\overline{P}) \notin coRE$. לכן $\overline{L(\overline{P})} \notin RE$, אבל בעצם $\overline{L(\overline{P})} = L(P)$ ולכן קיבלנו את המשפט. ■

<u>דוגמא לשימוש במשפט</u>:
נתבונן בשפה $L = \{\langle M\rangle : \forall w \in \Sigma^*,\ w \in L(M) \iff ww^Rww^R \in L(M)\}$. נרצה להראות שהשפה אינה כריעה, ובשביל זה צריך להשתכנע שהתכונה של M סמנטית ולא טריוויאלית. קל לראות שזו תכונה סמנטית, שכן היא מתייחסת רק לשפה של המכונה ולא למכונה עצמה. התכונה גם לא טריוויאלית, שהרי $\langle T_0\rangle \in L$. לכן לפי משפט רייס $L \notin R$, ולמעשה לפי הגרסה ההפוכה של הלמה, $L \notin RE$.

5.7 דקדוקים פורמליים

הגדרה 5.31 דקדוק פורמלי (formal grammar או context־sensitive grammar) הוא רביעייה $G = (V, \Sigma, R, S)$ כאשר V קבוצת משתנים, Σ א"ב של טרמינלים המקיים $\Sigma \cap V = \emptyset$, R אוסף כללי גזירה ו־ $S \in V$ משתנה התחלה; ובנוסף כלל גזירה הוא ביטוי מהצורה $x \to y$ כאשר $x \in (\Sigma \cup V)^* \cdot V \cdot (\Sigma \cup V)^*$ ו־ $y \in (\Sigma \cup V)^*$.

דוגמא לדקדוק פורמלי:

$$\begin{aligned} &\to S \\ S &\to aBc \\ C &\to b|Bc|\varepsilon \\ Bc &\to bC \end{aligned}$$

בדקדוק זה, אפשר לגזור את המילה ab באופן הבא: $S \to aBc \to abC \to ab$.

טענה 5.32 $L \in RE$ אסם קיים דקדוק פורמלי G כך ש־ $L(G) = L$.

הוכחה: (ר' תרגיל 8, שאלה 5) סקיצת הוכחה:
בהינתן דקדוק פורמלי ומילה, אנו יכולים לבדוק את כל אפשרויות הגזירה של המילה בדקדוק באמצעות BFS (כל התפצלות כי מקום שבו אפשר להשתמש ביותר מאשר כלל גזירה אחד). אם המילה נגזרת מהדקדוק, בסופו של דבר נגיע אליה ונגזור אותה.
בכיוון השני, בהינתן שפה הניתנת למניה רקורסיבית ומ"ט המזהה אותה, ניתן לבנות דקדוק פורמלי שפעולתו מחקה את מהלך הריצה של המכונה על מילת קלט כלשהי וניתן לגזור ממנו רק מילים שמתקבלות ע"י המכונה (ר' גם הטענה הבאה). ■

טענה 5.33 תהי $A_G = \{(\langle G\rangle, w) : \ w \in L(G)\}$. אזי $A_{TM} \leq_m A_G$.

הוכחה: בהינתן קלט $(\langle M\rangle, w)$ נבנה דקדוק G עם הכללים הבאים שגורמים לכך שתהליך הגזירה של G יחקה את תהליך הריצה של M על w:

$$S \ \rightarrow \ \#\overset{w_1}{\underset{q_0}{}}w_2w_3\dots w_n\#$$

כאשר כאן $\overset{w_1}{\underset{q_0}{}}$ משמעו שהמצב הנוכחי הוא q_0 והראש הקורא/כותב נמצא מעל w_1. עבור כלל מעבר מהצורה $\delta(q_1,a) = (q_2,b,R)$ נייצר לכל $c \in \Gamma$ את הכלל:

$$\overset{a}{\underset{q_1}{}}c \ \rightarrow \ b\overset{c}{\underset{q_2}{}}$$

וכן את הכלל

$$\overset{a}{\underset{q_1}{}}\# \rightarrow b\overset{\sqcup}{\underset{q_2}{}}$$

וכנ"ל עבור כללים שבהם הראש הקורא/כותב פונה שמאלה. כללי הגזירה האלה גורמים לכך שהדקדוק אחרי t צעדים גוזר בדיוק את המילה המייצגת את ריצתה של M על w אחרי t צעדים. אם M הגיעה למצב מקבל, היינו צריכים לגזור את $\#\sigma_1\sigma_2\dots\sigma_k\overset{\sigma}{\underset{q_{acc}}{}}\sigma_{k+1}\dots\sigma_m\#$ ל־ ε, ונעשה את זה על פי הכללים הבאים, לכל $\sigma, a \in \Gamma$:

$$\begin{aligned}\overset{\sigma}{\underset{q_{acc}}{}} \ &\rightarrow \ P\\ Pa \ &\rightarrow \ P\\ aP \ &\rightarrow \ P\\ P \ &\rightarrow \ \varepsilon\end{aligned}$$

כעת ברור שאפשר לגזור את ε אסם M מגיעה ל־ q_{acc}, כלומר $(\langle G\rangle, \varepsilon) \in A_G \iff (\langle M\rangle, w) \in A_{TM}$ ובזאת השלמנו את הרדוקציה. ■

5.8 בעיית ה־ PCP

הקלט הוא קבוצה של זוגות מחרוזות $(x_1,y_1),\dots,(x_n,y_n)$ מעל א"ב Σ. כשנצייר אותם, יהיה נוח לכתוב $\overset{x_i}{y_i}$. אוסף הזוגות נמצא בשפה PCP אם קיימת סדרה לא ריקה של אינדקסים (מותרות חזרות) $i_1,\dots,i_k \in \{1,\dots,n\}$ כך ש־ $x_{i_1}\cdots x_{i_k} = y_{i_1}\cdots y_{i_k}$.
אם כן,

$$PCP = \{(\langle x_i, y_i\rangle)_{i=1}^n : \ \exists(i_1,\dots,i_k) \quad x_{i_1}\cdots x_{i_k} = y_{i_1}\cdots y_{i_k}\}$$

הערה 5.34 קל לראות ש־ $PCP \in RE$ - נעבור על כל סדרות האינדקסים בסדר לקסיקוגראפי ונבדוק האם קיבלנו פתרון. אם יש פתרון - בסופו של דבר אנחנו נמצא אותו.

טענה 5.35 $PCP \notin R$.

הוכחה: נראה זאת באמצעות רדוקציה מ־ A_G, וכבר ראינו ש־ $A_{TM} \leq_m A_G$. המטרה היא שצירוף "אבני הדומינו" של PCP יהיה גזירה של מילה בדקדוק הפורמלי.
נתבונן בדוגמא של הדקדוק הפורמלי הבא:

$$\begin{aligned} \to S & \\ S &\to aSS|b \\ aS &\to b \end{aligned}$$

וגזירה לדוגמא בדקדוק זה: $S \to aSS \to bS \to bb$.
<u>ניסיון ראשון</u>: מה שמופיע בחלק התחתון בשלב הגזירה הבא צריך להופיע בחלק העליון בשלב הקודם. למשל, אם עד עכשיו גזרנו את α, אז השלב הבא יהיה $\begin{smallmatrix}\beta\\ \alpha\end{smallmatrix}$ אם אפשר לגזור $\alpha \to \beta$. במונחי הדוגמא מקודם, נשים לב למשל שעבור השלב $aSS \to bS$ שמיוצג ע"י $\begin{smallmatrix}b\\ aS\end{smallmatrix}$ אנו צריכים לבטא גם את ה־ S שנשאר. אז אפשר להשאיר גם בלוק מהצורה $\begin{smallmatrix}\sigma\\ \sigma\end{smallmatrix}$ לכל $\sigma \in \Sigma \cup V$, אבל זה בעייתי כי אז נוצר פתרון ל־PCP אפילו בנוכחות בלוק אחד כזה. נצטרך להימנע מהם.
<u>ניסיון שני</u>: הבלוקים יהיו כדלקמן:
1. לכל $\sigma \in \Sigma \cup V$ יהיה לנו הבלוק $\begin{smallmatrix}\sigma\\ \tilde{\sigma}\end{smallmatrix}$ והבלוק $\begin{smallmatrix}\tilde{\sigma}\\ \sigma\end{smallmatrix}$, כאשר נייחס ל־ $\tilde{\sigma}$ את אותה משמעות כמו ל־ σ מבחינת חוקי הדקדוק המקורי.
2. לכל כלל גזירה $\alpha \to \beta$ יהיה לנו הבלוק $\begin{smallmatrix}\beta\\ \tilde{\alpha}\end{smallmatrix}$ והבלוק $\begin{smallmatrix}\tilde{\beta}\\ \alpha\end{smallmatrix}$.
3. בלוק התחלה: $\begin{smallmatrix}\diamond S\#\\ \diamond\end{smallmatrix}$ כאשר $\diamond$ מסמן התחלה ו־ $\#$ מסמנת הפרדה בין שלבי הגזירה.
4. בלוק סיום: $\begin{smallmatrix}\square\\ w\#\square\end{smallmatrix}$ כאשר $\square$ מסמן סוף ו־ w היא מילת הקלט.

כדי לטפל בתווי $\#$, נאפשר בסוג הראשון של הבלוקים גם $\sigma = \#$. להלן דוגמא לגזירה של המילה bb בדקדוק הפורמלי שהראינו קודם:

$$\begin{array}{cccccccccccccc} \diamond S\# & \tilde{a}\tilde{S}\tilde{S} & \tilde{\#} & b & S & \# & \tilde{b} & \tilde{b} & \tilde{\#} & b & b & \# & \square \\ \diamond & S & \# & \tilde{a}\tilde{S} & \tilde{S} & \tilde{\#} & b & S & \# & \tilde{b} & \tilde{b} & \tilde{\#} & bb\#\square \end{array}$$

נשים לב ששלב הגזירה האחרון היה הכרחי כי אין לנו בלוק שבחלקו התחתון יש $\tilde{b}\tilde{b}$, אלא רק בלוק שיש בו bb. לכן אנו עושים עוד צעד העתקה קצת מיותר שכזה.
כעת יש לנו תהליך חישוב שבונה קלט ל־PCP מ־ $\langle G \rangle, w$. אם $w \in L(G)$ אז יש גזירה $S = \gamma_1 \to \ldots \to \gamma_m = w$ של המילה ב־ G. הפתרון PCP המתאים הוא $\square\#\gamma_m\#\ldots\#\gamma_m\#\tilde{\gamma_1}\#\gamma_2\#\ldots\#\gamma_m\#\gamma_m\#\square$, כאשר אם m אי־זוגי צריך בסוף $\diamond S\#\tilde{\gamma_1}\#\gamma_2\#\ldots\#\gamma_m\#\gamma_m\#\square$ כמו בגזירה מקודם.
להיפך, בהינתן פתרון לבעיית ה־PCP, ברור שבלוק ההתחלה צריך להיות ראשון ובלוק הסיום צריך להיות אחרון בהתאמת המחרוזות, וכל שלבי הביניים מתאימים לכללי גזירה של G. כלומר, במחרוזת פתרון ה־PCP, כל מעבר הוא גזירה חוקית ב־ G.
לכן $A_G \leq_m PCP$ כפי שרצינו. ■

5.9 פונקציות לא־חשיבות[5]

לפני שנעסוק בפונקציות, ננסה לחשוב על מספרים חשיבים. למשל, האם לכל מספר ממשי x קיימת תוכנית מחשב (או מכונת טיורינג) שפולטת את x? קל לראות משיקולי ספירה שהתשובה היא שלילית - קבוצת המספרים הממשיים אינה בת־מניה, ואילו קבוצת תוכניות המחשב - ואפילו קבוצת כל המחרוזות הסופיות - היא בת־מניה.
נרצה לראות שתי דוגמאות קונקרטיות לפונקציות לא־חשיבות $f : \mathbb{N} \to \mathbb{N}$.
1. נסדר את כל המ"ט לפי סדר לקסיקוגראפי של התיאור שלהן - $M_1, M_2, \ldots$. כעת נגדיר את הפונקציה $f(i) = 1$ אם M_i עוצרת על ε, ו־ 0 אחרת. כיוון שבעיית העצירה אינה כריעה, הפונקציה הזאת אינה חשיבה.
2. עבור א"ב סרט $\{0, 1, \sqcup\}$ נגדיר את הפונקציה $BB : \mathbb{N} \to \mathbb{N}$ כך ש־ $BB(k)$ הוא מספר ה־ 1־ים המקסימלי שיכול להיות על הסרט לאחר שמ"ט M כלשהי בעלת k מצבים עם א"ב סרט כנ"ל <u>עוצרת</u> בריצתה על ε. (כלומר, עוברים על כל המ"ט שיש להן k מצבים מעל הא"ב הנתון, ובודקים כמה 1־ים יש על הסרט כשהן מסיימות את ריצתן על ε.)
ראשית, ברור ש־ BB מונוטונית עולה ממש. בהינתן מ"ט שמדפיסה $BB(k)$ אחדות על הסרט בריצתה (העוצרת) על ε, נחליף את המצב המקבל שלה במצב נוסף שזז על הסרט ימינה עד שהוא מוצא משהו שאינו 1, ואז כותב 1 ועובר למצב המקבל. מכאן $BB(k+1) \geq BB(k) + 1 > BB(k)$.
נניח בשלילה ש־ BB חשיבה, ותהי H מכונת טיורינג המחשבת אותה. נניח בה"כ ש־ H מקבלת את הקלט שלה בבינארי ופולטת את הפלט שלה באונארי, כלומר מקבלת מספר בינארי k ועוצרת כשעל הסרט כתובות $BB(k)$ אחדות.
נניח שב־ H יש n מצבים. נבנה את המכונה M שבה $\lceil \log n \rceil + 1$ מצבים שתוכל לעצור על ε כאשר על הסרט שלה כתוב בבינארי $2n$. כעת נבנה את המכונה M' שבהינתן ε מתחילה בתור M ואחרי ש־ $2n$ נכתב על הסרט (בבינארי), היא מעבירה את

[5] החומר להלן לא נלמד במהלך הקורס, אלא מופיע בסיכום הרשמי של תרגול 8.

השליטה ל־ H שמדפיסה את $BB(2n)$. אבל זוהי סתירה, כי ל־ M' יש $n + \lceil \log n \rceil + 1$ מצבים והיא עוצרת עם $BB(2n)$ אחדות על הסרט, ולכן מקבלים ש־ $BB(n + \lceil \log n \rceil + 1) \geq BB(2n)$, וזו סתירה למונוטוניות (שהרי $2n > n + \lceil \log n \rceil + 1$).
(אפשר לחשוב על הנ"ל בתור תוכנית מחשב - אם יש פונקציה שאומרת כמה אחדות יכולה לפלוט כל פונקציה בעלת מספר מוגבל של תוים, אפשר לכתוב פונקציה חדשה שמעבירה לפונקציה המקורית מספר שגדול ממספר התוים בפונקציה המקורית + בפונקציה החדשה.)

חלק III
סיבוכיות

6 מחלקות זמן, P ו־ NP

בפרק זה נתחיל למדוד את משאבי החישוב של מכונות טיורינג, ונתעניין בשאלה האם אפשר להכריע בעיה מסוימת תוך זמן מוגדר היטב, ולא "סתם להכריע" אותה.

6.1 מחלקות זמן

הגדרה 6.1 תהי $L \in R$ ותהי M מ"ט המכריעה אותה. נאמר ש־ $t_M(n)$, הזמן שלוקח ל־ M לרוץ על קלט באורך n, הוא

$$t_M(n) = \max_{|x| \leq n,\ x \in \Sigma^*} \{\text{number of steps of } M \text{ before it stops on } x\}$$

הערה 6.2 כמובן, הגדרה זו תלויה במכונה - ורוצים לשאול מהו הזמן עבור מ"ט <u>אופטימלית</u> שמזהה את L. כמו כן, נשים לב שאפשר להגדיר את $t_M(n)$ על כל מכונת טיורינג שעוצרת תמיד, ולא רק על כזאת שמכריעה את L.

הגדרה 6.3 תהי $f : \mathbb{N} \to \mathbb{R}^+$ פונקציה מונוטונית עולה, ויהי Σ א"ב. אזי:

$$\text{TIME}[f(n)] = \{L \subseteq \Sigma^* :\ \exists M \text{ TM that decides } L,\ t_M(n) = \mathcal{O}(f(n))\}$$

במילים אחרות, $L \in \text{TIME}[f(n)]$ אם קיימת מכונת טיורינג דטרמיניסטית כך שלכל $w \in \Sigma^*$ המכונה עוצרת תוך $\mathcal{O}(f(|w|))$ צעדים, ומכריעה את L.

נתחיל במחלקה המתבקשת ביותר, $\text{TIME}[n]$. ברור ש־ $REG \subseteq \text{TIME}[n]$ שכן מכונת טיורינג היא בפרט אוטומט סופי דטרמיניסטי, ואוטומטים פועלים בזמן ליניארי. ההיפך, כמובן, לא ברור:

משפט 6.4 (ללא הוכחה) תהי $f : \mathbb{N} \to \mathbb{R}^+$ מונוטונית עולה כך ש־ $\lim_{n\to\infty} \frac{f(n)}{n \log n} = 0$ וגם $f(n) \geq n$. אזי $\text{TIME}[f(n)] = REG$.

טענה 6.5 תהי $L = \{w \in \{a,b\}^* :\ \#_a(w) = \#_b(w)\}$. אזי $L \in \text{TIME}[n \log n]$.

הוכחה: ראינו כבר ש־ $L \notin REG$ וקל לראות ש־ $L \in \text{TIME}[n^2]$. כדי להראות את החסם הדרוש צריך קצת להתחכם. למשל, הרעיון של למיין את המחרוזת הוא נחמד אבל במכונת טיורינג לא ניתן למיין בזמן $\mathcal{O}(n \log n)$ - למשל, פעולת swap בין שני אלמנטים לא תיקח זמן קבוע. במקום זה:
נעשה n שלבים שבכל אחד מהם נקטין את מספר ה־ a־ים פי שניים ואת מספר ה־ b־ים פי שניים, על ידי מעבר על המחרוזת משמאל לימין ומחיקת כל a שני וכל b שני, וכמובן זוכרים האם היה מספר זוגי כדי לוודא בסוף שהמספרים מסתדרים. אם מספר ה־ a־ים וה־ b־ים לא היה בעל אותה זוגיות, דוחים, ואחרת עושים סבב נוסף על המחרוזת. כל שלב לוקח $\mathcal{O}(n)$ זמן ולכן סך זמן הריצה הוא $\mathcal{O}(n \log n)$ כנדרש. ■

הגדרה 6.6

$$P = \bigcup_{k=1}^{\infty} \text{TIME}[n^k]$$

זוהי מחלקת השפות שניתנות להכרעה בזמן פולינומיאלי.

דוגמאות לבעיות ב־P: קשירות של גרף, רשת זרימה, הפיכות של מטריצה - כמעט כל מה שלמדנו באלגוריתמים.

הגדרה 6.7 נסמן ב־ $\mathrm{NTIME}[f(n)]$ את מחלקת השפות המקיימות את הדרישות של $\mathrm{TIME}[f(n)]$ עבור מכונת טיורינג לא דטרמיניסטית.
במילים אחרות, $L \in \mathrm{NTIME}[f(n)]$ אם קיימת מכונת טיורינג לא דטרמיניסטית אשר על $w \in L$ יש לה ריצה מקבלת תוך $\mathcal{O}(f(|w|))$ צעדים, ועל $w \notin L$ כל הריצות שלה דוחות תוך $\mathcal{O}(f(n))$ צעדים, ובפרט כולן עוצרות.

$$NP = \bigcup_{k=1}^{\infty} \mathrm{NTIME}[n^k]$$

היחסים בין P ל־ NP מאוד מתוחים, וכבר מזה 40 שנה לא ידוע האם $P \neq NP$.

הערה 6.8 נשים לב שאנו לא דורשים מהמכונה הלא דטרמיניסטית לעצור תוך $\mathcal{O}(f(|w|))$ צעדים בכל הריצות, אלא רק בחלק מהן.

טענה 6.9 $L \in NP$ אמם קיים קבוע k ומכונה לא דטרמיניסטית M כך ש־ M עוצרת על כל הקלטים תוך זמן $\mathcal{O}(n^k)$ ומכריעה את L.

הוכחה: אנו יודעים שקיימת מ"ט M' המקיימת את ההגדרה עם זמן ריצה $\geq c'n^{k'}$. נבצע סימולציה לא דטרמיניסטית למכונה זו במשך $c'n^{k'}$ צעדים. אם היא עדיין לא סיימה את כל ריצותיה, נדחה. אחרת, נענה כמוה. הסימולציה היא פולינומיאלית כנדרש. ■

הגדרה 6.10 נוסחא במשתנים בוליאניים מוגדרת באינדוקציה בקורס בלוגיקה. אנחנו נסתפק בהגדרה ע"י דוגמא, למשל $(\overline{x_1} \vee x_2) \wedge (x_3 \vee \overline{x_2} \vee \overline{x_4})$ זו נוסחא במשתנים בוליאניים.

טענה 6.11 $\mathrm{SAT} = \{\langle\varphi\rangle : \ \varphi \text{ is a satisfiable Boolean formula}\} \in NP$.

הוכחה: קל לראות שע"י מעבר על כל האפשרויות, $\mathrm{SAT} \in \mathrm{TIME}[2^n]$ כאשר n הוא מספר המשתנים הבוליאניים בנוסחא. כיצד ניתן להכריע את SAT בזמן פולינומי באמצעות מכונה לא דטרמיניסטית? ובכן, המכונה רושמת על הסרט באופן לא דטרמיניסטי אחד מ־ 2^n הצירופים שיש, וקוראת לפרוצדורה ליניארית שבודק האם ההשמה מספקת. אם יש השמה - המכונה תעצור ותחזיר את התשובה הנכונה, כלומר יש ריצה מקבלת, בזמן ליניארי. לכן $\mathrm{SAT} \in NP$. ■

6.2 הצגת NP באמצעות מוודאים

הגדרה 6.12 מוודא (verifier) לשפה L זו מכונת טיורינג דטרמיניסטית V כך ש:

$$L = \{x : \ \exists y : \ \langle x, y\rangle \in L(V)\}$$

כאשר y נקרא העד של x, וגם ש־ V עוצרת על כל הקלטים בזמן פולינומיאלי ב־ $|x|$.

דוגמאות למוודאים:
1. במקרה של SAT, העד של שייכות נוסחא ל־ SAT היא השמה של ערכי אמת לפסוקיות, והמוודא זוהי פרוצדורת הבדיקה שמוודאת שההשמה אכן מספקת.
2. נתבונן בשפה $\mathrm{HAMPATH} = \{\langle G\rangle : \ G \text{ contains a Hamiltonian path}\}$. כמובן יש מוודא לשפה - המוודא מקבל את הגרף ומסלול בגרף, ובודק האם זה אכן מסלול המילטוני. כלומר,

$$L(V) = \{\langle G, P\rangle : \ P \text{ is a Hamiltonian path in } G\}$$

כמובן, זמן הריצה בשני המקרים הוא פולינומיאלי באורך הקלט המקורי.

טענה 6.13 תהי L שפה. קיים מוודא V ל־ L אמם $L \in NP$.

הוכחה: ($\Rightarrow$) העד לשייכות L ל־ NP זוהי סדרת קונפיגורציות שמייצגת ריצה מקבלת של מ"ט לא דטרמיניסטית M המכריעה את L בזמן פולינומי. המוודא בודק שסדרת הקונפיגורציות היא חוקית ומסתיימת בקבלה. נשים לב שאורך סדרת הקונפיגורציות הוא פולינומי באורך הקלט כי כל קונפיגורציה היא בגודל פולינומי ויש מספר פולינומי של קונפיגורציות, ולכן גם המוודא פועל בזמן פולינומי. (אפשרות "קצרה" יותר: במקום סדרת קונפיגורציות, להציג רק את הבחירות שהתקבלו בכל מקום שהייתה למכונה אפשרות לא דטרמיניסטית לפעולה, כאשר הבחירה מוצגת כמספר למשל. בתרגול קראנו לזה ה"כתובת" של הריצה.)
($\Leftarrow$) יהי V מוודא פולינומי ב־ $|x|$ ונראה שיש N מ"ט לא דטרמיניסטית כך ש־ $L(N) = L$ המכריעה תוך $\mathcal{O}(n^k)$ צעדים. נניח ש־ V מכריע תוך $\mathcal{O}(|x|^k)$ צעדים. לכן אורך עד ל־ x יכול להיות לכל היותר $|y| = \mathcal{O}(|x|^k)$, כי אם העד המינימלי ארוך מזה לא ייתכן זמן ריצה קצר יותר, שהרי V צריך לקרוא את העד.
המכונה N תרשום y כלשהו על הסרט (אחד מכל העדים האפשריים באורך $\mathcal{O}(n^k)$) בקידוד $\langle x, y\rangle$ ותפעיל עליהם את המוודא V. זוהי ריצה דטרמיניסטית ופולינומיאלית ב־ $|x|$. ברור שאם קיים עד y, באחת הריצות נקבל; אחרת, אף ריצה לא תקבל וכל הריצות ידחו. ■

הערה 6.14 בהוכחה זו, וגם בהוכחות אחרות נשתמש בחופשיות ב־ $\mathcal{O}$. כמובן, כשהמכונה N צריכה לכתוב על הסרט עדים, היא צריכה לדעת בדיוק את הקבוע של cn^k.

6.3 בעיות קלאסיות ב־ NP

הבעיות הבאות הן בעיות קלאסיות ב־ NP. בהמשך נחזור אליהן כדי לראות שהן גם NP־שלמות.

1. קבוצה בת"ל - Independent Set - האם בגרף G קיימת קבוצה בת"ל בגודל k?
2. קליקה - Clique - האם בגרף G קיימת קליקה בגודל k?
3. מסלול המילטוני - Hamiltonian Path - האם בגרף G קיים מסלול המילטוני?
4. כיסוי ע"י קבוצות - Set Cover - נתון אוסף של ת"ק $A_1, \ldots, A_m \subseteq \{1, \ldots n\}$ ושואלים האם $\{1, \ldots, n\}$ מתקבל כאיחוד של $k \geq$ מהקבוצות $A_1, \ldots, A_m$?
5. כיסוי ע"י קודקודים - Vertex Cover - האם בגרף G ניתן לגעת בכל הצלעות בעזרת k קודקודים?
6. CNFSAT - האם נוסחא בוליאנית בצורת CNF (Conjunctive Normal Form) היא ספיקה? (נוסחא בצורת CNF היא נוסחא המחברת ב־ $\wedge$ בין פסוקיות בסוגריים שבתוכן יש $\vee$ בין מספר ליטרלים, למשל $(x_2 \vee \overline{x_1}) \wedge (\overline{x_3} \vee x_2 \vee x_4) \wedge x_1$.)

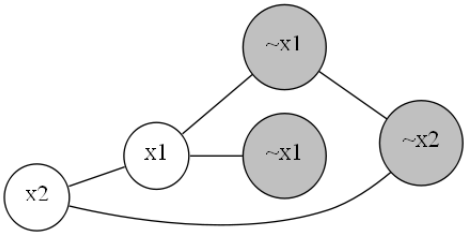
נראה, למשל, ש־ $\text{SetCover} \in NP$. יש $\binom{m}{k}$ אפשרויות לבחור k ת"ק מתוך m הקבוצות הנתונות, והבדיקה האם אוסף של k ת"ק מכסה את $\{1, \ldots, n\}$ לוקחת זמן ליניארי ב־ n. לכן המכונה הלא דטרמיניסטית יכולה פשוט לכתוב על הסרט צירוף כלשהו של $A_1, \ldots, A_m$ בגודל k, ואז להריץ מוודא דטרמיניסטי ולינארי שבודק את הכיסוי.
במובן מסוים, כל הבעיות האלה הן ב־ NP כי בתנאי שיש פתרון - "קל יחסית" (בזמן פולינומיאלי) לבדוק שזהו הפתרון הנכון. עבור אף אחת מהבעיות הנ"ל לא ידוע האם "קל יחסית" (בזמן פולינומיאלי) למצוא את הפתרון.

6.4 רדוקציות מיפוי פולינומיאליות, קושי ושלמות ב־ NP

הגדרה 6.15 יהיו $L_1, L_2 \subseteq \Sigma^*$. רדוקציית מיפוי פולינומיאלית (או רדוקציית Karp) מ־ L_1 ל־ L_2 היא רדוקציית מיפוי מ־ L_1 ל־ L_2 שרצה בזמן פולינומיאלי. אם יש רדוקציה כזאת, נכתוב $L_1 \leq_p L_2$.

טענה 6.16 $\text{CNFSAT} \leq_p \text{IndependentSet}$.

הוכחה: בהינתן נוסחא, נייצר גרף שבו קודקוד לכל הופעה של ליטרל בנוסחא. נחבר בצלע בין שני ליטרלים אם הם באותה פסוקית, או אם הם סותרים. נקבע את k להיות מספר הפסוקיות בנוסחא.
לדוגמא, עבור הנוסחא $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge \overline{x_1}$ נייצר את הגרף הבא:



קל לראות שהתהליך הזה הוא פולינומיאלי, אבל מדוע זאת רדוקציה? ובכן, בתוך כל פסוקית חיברנו את כל הקודקודים, כך שאם בחרנו קודקוד מתוך פסוקית, שאר הקודקודים בפסוקית לא יהיו באנטי־קליקה, ולכן מספר הקודקודים באנטי־קליקה לא יעלה על מספר הפסוקיות. בנוסף, חיברנו את הליטרלים הסותרים ולכן באנטי־קליקה לא תהיה סתירה בין הליטרלים. ■

הגדרה 6.17 תהי L שפה כך שלכל $L' \in NP$ מתקיים $L' \leq_p L$. אז נגיד ש־ L היא NP־קשה.
אם בנוסף $L \in NP$ אז נגיד ש־ L היא NP־שלמה.
נסמן ב־ NPC את אוסף השפות ה־ NP־שלמות.

משפט 6.18 (Cook־Levine, 1972) $\text{CNFSAT} \in NPC$. (הוכחה בהמשך.)

מסקנה 6.19 $\text{IndependentSet} \in NPC$.

טענה 6.20 $\text{CLIQUE} \in NPC$.

הוכחה: מספיק להראות ש־ $\text{IndependentSet} \leq_p \text{CLIQUE}$. ואמנם, הרדוקציה היא פשוט לפלוט את הגרף המשלים (בצלעות). ■

טענה 6.21 $\text{VC} \in NPC$.

הוכחה: נשים לב שאם $U \subseteq V$ אז הוא כיסוי בקודקודים של הצלעות אסם $V \setminus U$ קבוצה בת"ל. אכן, אם $u, v \in V \setminus U$ ויש ביניהם צלע, הרי שהיא לא מכוסה. לכן בגרף יש כיסוי בקודקודים בגודל k אסם יש בו קבוצה בת"ל בגודל $|V| - k$ ולכן הרדוקציה היא $\langle G, k \rangle \to \langle G, |V| - k \rangle$. ■

6.5 משפט Cook־Levin

הגדרה 6.22

$$\text{Bounded}A_{NTM} = \{(\langle M \rangle, w, 1^t) : \quad \exists y,\ |y| \leq t,\ M \text{ accepts } w\#y \text{ in time } \leq t\}$$

הערה 6.23 קל לראות ש־ $\text{Bounded}A_{NTM} \in NP$, שכן מכונה לא דטרמיניסטית יכולה לנחש את y שהוא באורך t לכל היותר, ולבצע סימולציה של M במשך $t \geq$ צעדים, שזה ייקח זמן פולינומיאלי ב־ t כנדרש. זו גם הסיבה שבגללה אנו מעבירים את t בייצוג אונארי - כדי לקבל זמן ריצה פולינומיאלי ב־ t.

טענה 6.24 $\text{Bounded}A_{NTM} \in NPC$.

הוכחה: תהי $L \in NP$. ראינו שיש מ"ט דטרמיניסטית M וקבועים c, k כך שלכל $w \in \Sigma^*$ מתקיים ש־ $w \in L$ אסם קיים עד y כך ש־ M מקבלת את $w\#y$ תוך לכל היותר cn^k צעדים, כאשר $|w| = n$.
בהינתן קלט w, הפלט של הרדוקציה הוא $(\langle M \rangle, w, 1^t)$ כאשר M המכונה הנ"ל ו־ $t = cn^k$. הרדוקציה לוקחת זמן פולינומיאלי, וכמובן $w \in L$ אסם $(\langle M \rangle, w, 1^t) \in \text{Bounded}A_{NTM}$ לפי הגדרתה. ■

הגדרה 6.25 בהינתן $x_1, \ldots, x_k$ משתנים, $\mathcal{A}$ קבוצת ההשמות הבוליאניות למשתנים, $p : \mathcal{A} \to \{0,1\}$ נקראת <u>פרדיקט</u> (predicate) על המשתנים $x_1, \ldots, x_k$.

<u>דוגמאות לפרדיקטים</u>:
1. הנוסחא $x_4 \wedge \overline{x_1} \wedge (x_1 \vee x_2 \vee x_3 \vee \overline{x_4})$ מגדירה פרדיקט, כי עכשיו לכל השמה של ערכים בוליאניים למשתנים אנו מקבלים ערך אמת עבור הנוסחא. בעצם, כל נוסחת CNF ובכלל כל נוסחא בוליאנית מגדירה פרדיקט.
2. הפסוק "אם x_1 או x_2 אז x_3 וגם x_4" מגדיר פרדיקט.
3. כל פונקציה $f : \{0,1\}^k \to \{0,1\}$ מגדירה פרדיקט p_f על $x_1, \ldots, x_k$ באופן ברור.

טענה 6.26 לכל $f : \{0,1\}^k \to \{0,1\}$ הפרדיקט p_f ניתן להצגה ע"י נוסחת CNF.

הוכחה: דרך אחת לעשות זאת - בהינתן $\alpha = (\alpha_1, \ldots, \alpha_k) \in \{0,1\}^k$ נגדיר $D_\alpha = l_1 \wedge \ldots \wedge l_k$ כאשר $l_i = \begin{cases} x_i & \alpha_i = 1 \\ \overline{x_i} & \alpha_i = 0 \end{cases}$. כעת $p_f = \bigvee_{\alpha : f(\alpha) = 1} D_\alpha$.

הבעיה היא שקיבלנו נוסחת DNF ולא נוסחת CNF, ונתקן את זה ע"י הגדרת $C_\alpha = l_1 \vee \ldots \vee l_k$ כאשר $l_i = \begin{cases} \overline{x_i} & \alpha_i = 1 \\ x_i & \alpha_i = 0 \end{cases}$ ואז $p_f = \bigwedge_{\alpha : f(\alpha) = 0} C_\alpha$, וזו כבר נוסחת CNF. ■

מסקנה 6.27 אם רוצים נוסחת CNF, אפשר להסתפק בגימום של הרבה פרדיקטים שכל אחד מהם הוא ב־ CNF. עדיף שכל פרדיקט יהיה מעל מספר קטן מאוד של משתנים, שהרי הצגת CNF של פרדיקט מעל k משתנים עשויה להיות באורך $\mathcal{O}(k \cdot 2^k)$.

משפט 6.28 $\text{Bounded}A_{NTM} \leq_p \text{CNFSAT}$ (משפט Cook־Levin, 1972)

הוכחה: בהינתן $(\langle M\rangle, w, 1^t)$ הרדוקציה תבנה קבוצת משתנים בוליאניים ונוסחת CNF φ מעליהם. לכל ריצה באורך $t \geq$ של M תהיה השמה יחידה למשתנים ה"מקודדת" את הריצה. ההשמות היחידות ש־ φ מקבלת יהיו ייצוגים של ריצות מקבלות על $w\#y$ כאשר $|y| \leq t$.

כאשר מייצגים ריצה של M ע"י השמה, נרצה לקודד את מצב הסרט, את המצב של המכונה, ואת המיקום של הראש הקורא/כותב. אין טעם להסתכל על מקומות בסרט מעבר לתא t או מעבר לתא $-t$. כמו כן, נסכים בינינו שאם הגענו למצב מקבל/דוחה תוך פחות מ־ t צעדים, אז נחזור על אותה קונפיגורציה עד שיעברו t צעדים. לכל "עובדה" מהריצה הזאת יהיה לנו משתנה בוליאני:

1. משתני הסרט -

$$\forall s \in \{0,\ldots,t\},\ i \in \{-t,\ldots,t\},\ a \in \Gamma \cup \{\sqcup\}: \quad X_{s,a,i}$$

כאשר המשתנה $X_{s,a,i}$ מייצג את הטענה: "בזמן s במקום ה־ i על הסרט כתובה האות a".

2. מיקום הראש -

$$\forall s \in \{0,\ldots,t\},\ i \in \{-t,\ldots,t\}: \quad h_{s,i}$$

כאשר המשתנה $h_{s,i}$ מייצג את הטענה: "בזמן s הראש נמצא במקום ה־ i".

3. משתני המצב -

$$\forall s \in \{0,\ldots,t\},\ q \in Q: \quad g_{s,q}$$

כאשר המשתנה $g_{s,q}$ מייצג את הטענה: "בזמן s מצב המכונה הוא q".

כעת נבנה את הנוסחא שלנו, שתהיה מורכבת מגימום של פרדיקטים:

1. מניעת ריבוי - הראש נמצא במקום אחד בלבד, בכל תא של הסרט כתובה רק אות אחת, המכונה נמצאת במצב אחד בלבד, וכו'. למשל עבור מצב הסרט: $\forall s,i,a \neq b: \quad \overline{X_{s,a,i}} \vee \overline{X_{s,b,i}}$, ועבור מצב המכונה: $\forall s, q_1 \neq q_2: \quad \overline{g_{s,q_1}} \vee \overline{g_{s,q_2}}$.
2. קיום מצבים - יש מקום שבו הראש נמצא, וכו'. למשל, עבור מצב הסרט: $\forall s,i: \quad \bigvee_{a\in\Gamma\cup\{\sqcup\}} X_{s,a,i}$.
3. כללי מעבר - עבור כל שוויון מהצורה $\delta(q_1,a) = (q_2,b,R)$ נגדיר את הפרדיקט לכל s,i: "אם $X_{s,a,i}$ וגם $h_{s,i}$ וגם g_{s,q_1} אז $X_{s+1,b,i}$ וגם $h_{s+1,i+1}$ וגם g_{s+1,q_2}". נשים לב שהפרדיקט הוא על מספר קבוע של משתנים, ולכן תרגומו ל־ CNF לא יגרום להתפוצצות אקספוננציאלית באורך הקלט (גודל הנוסחא). כמו כן, צריך להוסיף כאן את ההתייחסות לכך שאם כבר הגענו למצב מקבל/דוחה, אנו שומרים על אותו מצב כמו קודם - זו פשוט תוספת להגדרת δ.
4. תנאי התחלה - המילה נמצאת על הסרט: $\forall 0 \leq i \leq |w|-1: \quad X_{0,w_i,i}$ וה־ $\#$ נמצאת על הסרט: $X_{0,\#,|w|}$; מצב ההתחלה - $h_{0,0}$ וגם g_{0,q_0}. באופן דומה יש לפרט את המשתנים לגבי y.
5. הריצה מקבלת - כלומר $g_{t,q_{acc}}$.

ברור שקיבלנו רדוקציה פולינומיאלית, ולכן השלמנו את ההוכחה - CNFSAT היא NP־שלמה. ■

6.6 עוד בעיות NP־שלמות

נראה מספר דוגמאות לרדוקציות שבסופן נשתכנע שכל השפות שראינו הן NP־שלמות.

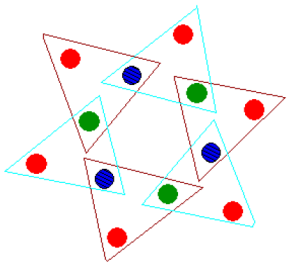
1. השפה 3DMatching היא כל הקלטים המתארים שלוש קבוצות שוות גודל, ושלשות המוגדרות על איברי הקבוצות, כך שקיים זיווג תלת־מימדי של איברי שלוש הקבוצות (כלומר, הכללה של זיווג בגרף דו"צ). בהינתן X,Y,Z שלוש קבוצות שגודל כל אחת m, ו־ $T \subseteq X \times Y \times Z$ אנו מחפשים m שלשות כך שכל איבר בכל קבוצה משתתף בשלשה אחת בדיוק.

קל לראות ש־ $\text{3DM} \in NP$ כי יש מוודא פולינומיאלי לזיווג מושלם. נראה שהבעיה קשה ע"י רדוקציה מ־ 3SAT (בהמשך נראה ש־ $\text{3SAT} \leq_p \text{CNFSAT}$[6]). בהינתן קלט של 3SAT - נוסחא φ בעלת k פסוקיות ו־ n משתנים - נבנה שני סוגים של gadget־ים עבור הבעיה החדשה: הסוג הראשון עבור משתנים, כדי לוודא שההשמה של ערכי האמת היא חוקית, והסוג השני הוא עבור פסוקיות, כדי לוודא שההשמה היא מספקת.

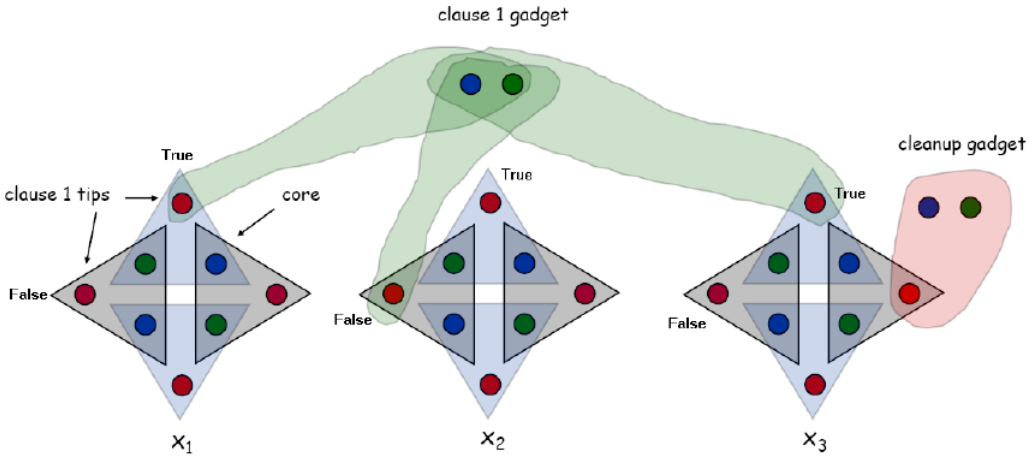
- לכל משתנה x, ניצור gadget כך: k נקודות ב־ X, k נקודות ב־ Y ו־ $2k$ נקודות ב־ Z, ונחבר אותם בשלשות כמו בציור[7]:

[6] חשוב לציין שיש סיבה לבחירה ב־ 3. למשל, אם מצמצמים את מספר הליטרלים בכל פסוקית ל־ 2, מקבלים שפה שניתן להכריע בקלות בזמן פולינומי (ע"י בדיקה שהנוסחא לא מכילה סתירה). ר' תרגיל 9, שאלה 4.

[7] הציורים נגנבו ללא בושה מסיכום תרגול 10.



- בכל gadget של משתנה x יש לנו $2k$ נקודות ב־ Z. נסמן אותן $\{z_1, z_2, \ldots, z_{2k}\}$ ונייחס להן את המשמעות הבאה: z_1 יקושר לעובדה ש־ x מקבל ערך $\mathbb{T}$ בפסוקית c_1, z_2 יקושר לעובדה ש־ x מקבל ערך $\mathbb{F}$ בפסוקית c_1, וכן הלאה. לבסוף z_{2_k} יקושר לעובדה ש־ x מקבל ערך $\mathbb{F}$ בפסוקית c_k.
- לכל פסוקית c_i, נוסיף שתי נקודות $x_i \in X$ ו־ $y_i \in Y$. לכל ליטרל l המופיע ב־ c_i נוסיף שלשה (x_i, y_i, z_j) כאשר z_j מקושר (מהסעיף הקודם) לעובדה ש־ l מקבל את הערך המתאים בפסוקית c_i. כלומר, אם למשל $l = \overline{a}$, אז z_j יהיה הקודקוד המקושר לעובדה ש־ a מקבל ערך $\mathbb{F}$ בפסוקית c_i. למשל, עבור הפסוקית $c_1 = x_1 \vee \overline{x_2} \vee x_3$ נקבל את המצב (החלקי) הבא:



- לבסוף, יש לנו יותר איברים ב־ Z מאשר בקבוצות האחרות. נוסיף נקודות נוספות ב־ X וב־ Y המחוברות בשלשות לכל איברי Z. יש לנו $(n-1)k$ שלשות כאלה.

ברור שתהליך הבנייה הוא פולינומיאלי, ונותר להראות שזוהי רדוקציה. מהשמה מספקת של נוסחא, לכל משתנה x שקיבל את הערך $\mathbb{T}$ אנו נבחר את השלשות שמכילות את ה־ z_i־ים כך ש־ z_i מקושר לעובדה ש־ x קיבל את הערך $\mathbb{F}$ בנוסחא (הפוך מההשמה), מה שמשאיר את ה־ z_i־ים המקושרים לעובדה ש־ x קיבל את הערך $\mathbb{T}$ פנויים. כעת מכל פסוקית נבחר משתנה כלשהו שמספק אותה, והשלשה שמחברת אותו עם ה־ z_j המתאים לו, שהוא בהכרח נותר פנוי.

להיפך, בהינתן זיווג בגרף, כל המעגלים הפנימיים ב־ gadgets של המשתנים מכוסים, וזה אפשרי רק אם בחרנו את כל השלשות הזוגיות או כל השלשות האי־זוגיות. מהזוגיות מתקבלת ההשמה למשתנים. גם כל ה־ gadgets של הפסוקיות מכוסים, ושוב זה אפשרי רק אם ה־ z־ים המתאימים פנויים. לכן התאמה משרה השמה מספקת.

2. כעת נראה ש־ $3\text{DM} \leq_p \text{SubsetSum}$. ניזכר ש־

$$\text{SubsetSum} = \{\langle s_1, \ldots, s_n, t\rangle : \ \exists I \subseteq \{1, \ldots, n\}, \ \sum_{i \in I} s_i = t\}$$

כאשר כל המספרים הם טבעיים. נראה כיצד הרדוקציה פועלת על הקבוצות

$$X = \{x_0, \ldots, x_{m-1}\}, \quad Y = \{y_0, \ldots, y_{m-1}\}, \quad Z = \{z_0, \ldots, z_{m-1}\}, \quad T = \{t_0, \ldots, t_n\} \subseteq X \times Y \times Z$$

לשלשה (x_i, y_j, z_k) נתאים מספר טבעי בעל $3m$ ספרות בבסיס b המחולק לשלושה חלקים, כל אחד בגודל m, שיש בו 1 במקומות $i,\ m+j,\ 2m+k$ ובשאר המקומות 0־ים. פורמלית,

$$(x_i, y_j, z_k) \longmapsto b^k + b^{m+j} + b^{2m+i}$$

כך מהקבוצה T אנו מייצרים את המספרים $s_1, \ldots, s_n$. בנוסף נייצר את המספר $t = 1 \ldots 1$ ($3m$ ספרות) בבסיס b. אין ספק שתהליך זה הוא פולינומיאלי, וכעת נראה שזו באמת רדוקציה.

בהינתן זיווג לבעיית 3DM אנו יודעים שיש ת"ק של T כך שעל כל X, Y, Z עוברים בדיוק פעם אחת על כל איבר. אם מסכמים את ה־s_i־ים הרלוונטיים לאותה ת"ק של T, נקבל כמובן את $1 \ldots 1$ ($3m$ ספרות). להיפך, בהינתן ת"ק של $s_1, \ldots, s_n$ המסתכמת ל־ T, נרצה למצוא זיווג. נגדיר כעת ש־ $b = n + 1$ וכעת ברור שלא יכול להיות נשא בחיבור, כי צריך לחבר לפחות $n + 1$ 1־ים כדי לקבל נשא בחיבור בבסיס $n + 1$. כלומר, כל ספרה מסתכמת לחוד וכל ספרה מסתכמת ל־ 1, ולכן יש בדיוק שלשה אחת שנוגעת בכל איבר מ־ X, Y, Z כפי שרצינו.

3. נראה ש־ $\text{3SAT} \leq_p \text{CNFSAT}$ כדי להשלים את העבודה. ב־ SAT יש לנו פסוקיות $\varphi = c_1 \wedge \ldots \wedge c_k$ אבל כל פסוקית מהווה איווי של מספר כלשהו של ליטרלים, $c_i = l_1^i \vee \ldots \vee l_{m_i}^i$. אנחנו רוצים רדוקציה לנוסחא שבה רק שלושה ליטרלים, ונעשה את זה עבור כל פסוקית בנפרד.

- אם מספר הליטרלים $m_i = 1$ נפלוט את הפסוקית $c_i' = l_1^i \vee l_1^i \vee l_1^i$ וכמובן שמרנו על ספיקות.
- אם $m_i = 2$ נפלוט פסוקית עם שכפול של אחד הליטרלים: $c_i' = l_1^i \vee l_2^i \vee l_2^i$.
- אם $m_i = 3$ נפלוט את הפסוקית כפי שהיא.
- אם $m_i > 3$ נפצל את c_i ל־ $m_i - 2$ פסוקיות עם משתנים חדשים מקשרים באופן הבא:

$$\begin{aligned}
c_i^1 &= l_1^i \vee l_2^i \vee z_1^i \\
c_i^2 &= \overline{z_1^i} \vee l_3^i \vee z_2^i \\
&\vdots \quad \vdots \quad \vdots \\
c_i^{m_i-3} &= \overline{z_{m_i-4}^i} \vee l_{m_i-2}^i \vee \overline{z_{m_i-3}^i} \\
c_i^{m_i-2} &= \overline{z_{m_i-3}^i} \vee l_{m_i-1}^i \vee l_{m_i}^i
\end{aligned}$$

כעת כדי להפוך השמה של CNF להשמה של 3CNF נצטרך לתת ערכי אמת לכל ה־ z־ים שיצרנו. לכל ה־ z־ים שלפני הליטרל הראשון שקיבל $\mathbb{T}$ ניתן את הערך $\mathbb{T}$, ועל כל אלה שאחריו - ניתן $\mathbb{F}$. להיפך, מהשמה של 3CNF צריך להראות השמה של CNF. אם z_i^1 הוא $\mathbb{F}$ אז בפסוקית הראשונה אחד הליטרלים הוא $\mathbb{T}$, ואם $z_{m_i-3}^i$ הוא $\mathbb{T}$ אז בפסוקית האחרונה אחד הליטרלים הוא $\mathbb{T}$. אם הראשון $\mathbb{T}$ והאחרון $\mathbb{F}$ אז יש מעבר מ־ $\mathbb{T}$ ל־ $\mathbb{F}$ איפשהו, למשל ב־ c_i^2, ואז שם $z_1^i = \mathbb{T}$ ו־ $z_2^i = \mathbb{F}$ אז בהכרח $l_3^i = \mathbb{T}$, וכן הלאה.

4. מסקנה מכל הנ"ל:

$$\text{CNFSAT} \leq_p \text{3SAT} \leq_p \text{3DM} \leq_p \text{SubsetSum}$$

ולכן כל הבעיות האלה הן NP־שלמות.

5. השפה $DS = \{\langle G, k\rangle : \ G \text{ has a dominating set of size at most } k\}$ היא NP־שלמה. ראינו זאת בתרגיל 9, שאלה 2 באמצעות רדוקציה מ־ VC. את כל קודקודי הגרף המקורי נשאיר כפי שהם, ונוסיף קודקוד v_e לכל צלע e בגרף המקורי. כמו כן, נייצר צלעות בין כל קודקודי הגרף המקורי, וצלעות בין כל v_e לקודקודים של הצלע e בגרף המקורי. גודל ה־ DS שמחפשים הוא כגודל ה־ VC.

6. השפה UHAMPATH (גרפים <u>לא מכוונים</u> שיש בהם מסלול המילטוני) היא NP־שלמה. ראינו זאת בתרגיל 10, שאלה 2 באמצעות רדוקציה מ־ HAMPATH (עבור גרפים מכוונים). במהלך הרדוקציה, יצרנו שלושה קודקודים v_{in}, v_{mid}, v_{out} עבור כל קודקוד של הגרף המקורי, חיברנו צלעות (v_{in}, v_{mid}), (v_{mid}, v_{out}) ודאגנו לכך שהצלעות המכוונות יהפכו לצלעות מן הצורה (v_{out}, u_{in}). תודות לבנייה הזאת, אנו מבטיחים שמסלול המילטוני בגרף החדש קיים אסם קיים מסלול המילטוני בגרף המקורי.

7. השפה NOTALLSAT של נוסחאות ה־ 3CNF כך שקיימת השמה שבכל פסוקית משאירה ליטרל אחד לא מסופק וליטרל אחד מסופק היא NP־שלמה. ראינו זאת בתרגיל 10, שאלה 3 באמצעות רדוקציה מ־ 3SAT. כל פסוקית מהצורה $(a_i \vee b_i \vee c_i)$ הופכת ל־

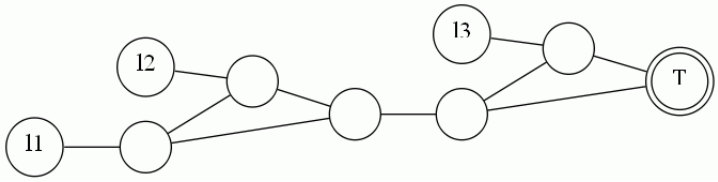
$$(a_i \vee b_i \vee x_i) \wedge (\overline{x_i} \vee c_i \vee y_i) \wedge (x_i \vee y_i \vee z)$$

כאשר z הוא משתנה חדש המשותף לכל הפסוקיות, ו־ x_i, y_i ספציפיים עבור כל פסוקית. לנוסחא המקורית יש השמה מספקת אסם לנוסחא החדשה יש השמה המקיימת את הדרישות מ־ NOTALLSAT.

8. השפה $\text{MAXCUT} = \{\langle G,k\rangle : \ G \text{ has a cut of size at least } k\}$ היא NP-שלמה. ראינו זאת בתרגיל 10, שאלה 3 באמצעות רדוקציה מ- NOTALLSAT. בגרף G החדש: לכל ליטרל יוצרים קודקוד, ובין כל שני ליטרלים המופיעים באותה פסוקית יוצרים צלע. בנוסף, על כל הופעה של משתנה בנוסחא, יוצרים את הקודקודים $x, \overline{x}$ ומחברים ביניהם בצלע. בסה"כ בגרף $6m$ צלעות והפלט של הרדוקציה הוא $\langle G, 5m\rangle$.

קל לראות שהשמה מתאימה מייצרת חתך בגודל $5m$ (צדדי החתך נקבעים לפי ערך האמת של הליטרלים), כי בכל פסוקית יש שתי צלעות בחתך וכל זוג $x, \overline{x}$ שהוספנו תורם עוד צלע. להיפך, $x, \overline{x}$ בהכרח נמצאים בשני צדדים שונים של החתך - אם מעבירים צד את זה שיש לו פחות צלעות בחתך, במקרה הטוב לא מרוויחים שום דבר.

9. השפה $\text{3COLOR} = \{\langle G\rangle : \ \text{there is a 3-coloring of } G\}$ היא NP-שלמה. ראינו זאת בתרגיל 11, שאלה 1 באמצעות רדוקציה מ- 3SAT. אנו יוצרים שלושה קודקודים המייצגים את הצבעים $\mathbb{T}, \mathbb{F}, K$ ומחברים אותם זה לזה - כך מובטח שהם ייצבעו בשלושה צבעים. כל משתנה ושלילתו מחברים ביניהם ול- K - כך מובטח שהם ייצבעו ב- $\mathbb{T}, \mathbb{F}$ מה שישרה את ההשמה. כדי להבטיח את התקינות לגבי הפסוקיות, מהפסוקית $l_1 \vee l_2 \vee l_3$ מייצרים gadget מהצורה:



ע"י בדיקה מייגעת של 7 מקרים מקבלים שזו אכן רדוקציה.

גם הבעיה $\text{COLOR} = \{\langle G,k\rangle : \ \text{there is a } k\text{-coloring of } G\}$ היא NP-שלמה באופן טריוויאלי, וגם : $k\text{-COLOR} = \{\langle G\rangle : \ k > 3, \ \text{there is a } k\text{-coloring of } G\}$ היא NP-שלמה באמצעות רדוקציה מ- 3COLOR ע"י הוספת $k-3$ קודקודים המחוברים לעצמם ולכל קודקודי הגרף המקורי.

6.7 בעיות חיפוש

לעתים התשובה לשאלה צריכה להיות לא בוליאנית אלא ממש "להראות משהו". למשל, לא רק להגיד שקיים זיווג בגרף, אלא להצביע על הזיווג הזה; לא רק להגיד שקיימת קליקה בגודל k, אלא ממש להצביע עליה. בעיות חיפוש הן לא קלות יותר מבעיות ההכרעה המתאימות, אבל בדרך כלל הן גם לא יותר קשות.

1. נניח שאנו מחפשים את גודל הקליקה המקסימלית בהינתן קופסה שחורה - מ"ט פולינומיאלית A - שמכריעה את CLIQUE. אפשר לעשות חיפוש בינארי, או סתם לעבור סדרתית על המספרים - עד שמגלים את גודל הקליקה המקסימלית k, ובכל מקרה גודל הקליקה לא יעלה על מספר הקודקודים. כדי למצוא את הקליקה עצמה, בודקים לכל $v \in V$ האם ב- $G \setminus \{v\}$ עדיין יש קליקה בגודל k. אם כן, אפשר להשמיט את v - וכך בסוף מגיעים לקליקה עצמה.

2. נניח שאנו מחפשים השמה מספקת של נוסחת CNF, בהינתן מ"ט פולינומיאלית A שמכריעה את CNFSAT (לחלופין יכולנו להניח, כמו בתרגיל 10, ש- $P = NP$). נפעל כך: ניתן ל- x_1 ערך $\mathbb{T}$ ונבדוק בעזרת A האם הנוסחא החדשה ספיקה. אם כן - נמשיך ל- x_2. אם לא - ניתן ל- x_1 ערך $\mathbb{F}$ וכן הלאה.

3. נניח שאנו מחפשים חתך בגודל מקסימלי בגרף, בהינתן מ"ט פולינומיאלית שמכריעה את MAXCUT (תרגיל 10, שאלה 4). תחילה נמצא את גודל החתך המקסימלי, k - למשל ע"י חיפוש ליניארי. לאחר מכן, עלינו לבדוק עבור כל צלע האם הסרתה מהגרף משפיעה על גודל החתך המקסימלי, ולחלק את הקודקודים בצדי הצלע בהתאם.

6.8 סגירות של NP ו- coNP

טענה 6.29 אם $L_1, L_2 \in NP$ אזי:

1. $L_1 \cup L_2 \in NP$
2. $L_1 \cap L_2 \in NP$
3. $L_1^* \in NP$

הוכחה: (סקיצה) 1. ננחש באופן לא דטרמיניסטי באיזו שפה הקלט נמצא ונריץ את המכונה המתאימה.
2. באמצעות מוודא - מקבלים שני עדים, ומוודאים את שניהם באמצעות שני המוודאים.
3. בהינתן מילה, נבקש מהעד את החלוקה לתת-מילים ועדים לגבי כל תת-מילה. ■

הערה 6.30 אבל, $L_1 \in NP \wedge L \subseteq L_1 \not\Rightarrow L \in NP$, למשל עבור $L_1 = \emptyset$ ו- $L = A_{TM}$. באופן דומה, $L_1 \in NP \wedge L \subseteq L_1 \not\Rightarrow L \in NP$, למשל עבור $L_1 = \Sigma^*$ ו- $L = A_{TM}$.

יתר על כן, לא ידוע האם $L \in NP \implies \overline{L} \in NP$. לכן:

הגדרה 6.31 $coNP = \{L : \ \overline{L} \in NP\}$, כלומר מחלקת השפות שהמשלים שלהן ב- NP.

לדוגמא, השפה $\text{PRIME} = \{\langle i\rangle : \ i \in \mathbb{N} \text{ is prime}\}$ היא ב- $coNP$, שכן העד לכך ש- $\langle k\rangle \notin \text{PRIME}$ יכול להיות מחלק שלו (שהוא בהכרח קטן ממנו). אגב, הוכיחו ב- 2002 ש- $\text{PRIME} \in P$.

הערה 6.32 יש בעיות שיודעים שהן ב־ $NP \cap coNP$ אבל לא יודעים האם הן ב־ P. לא ידועים גם היחסים בין NP ל־ $coNP$. מה שכן ברור זה ש־ $P \subseteq NP \cap coNP$, כי P סגורה למשלים.

טענה 6.33 אם $L_1 \in coNP$ ו־ $L_2 \leq_p L_1$, אזי $L_2 \in coNP$.

הוכחה: נובעת מיידית מכך שאם $A \leq_p B$ אז $\overline{A} \leq_p \overline{B}$. ■

הגדרה 6.34 אם $L \in coNP$ ולכל $L' \in coNP$ מתקיים $L' \leq_p L$ אזי נאמר ש־ L היא $coNP$־שלמה.

טענה 6.35 1. L היא $coNP$־שלמה אסם $\overline{L}$ היא NP־שלמה.
2. אם L היא $coNP$־שלמה ו־ $L \in NP$, אז $NP = coNP$.

הוכחה: תרגיל 11, שאלה 3. (זה ממש קל.) ■

<u>הפילוסופיה של coNP ו־ IP</u>:
בהינתן טענה מתמטית P ומקום 1^t לכתיבת ההוכחה שלה, ברור שבהינתן הוכחה אנו יכולים לוודא בזמן פולינומי שאכן מדובר בהוכחה כשרה. לכן הבעיה של מציאת הוכחה באורך t לטענה המתמטית P היא ב־ NP, ומסתבר שהיא גם NP־שלמה. זו אחת הסיבות שאנו מאמינים ש־ $P \neq NP$ - אם $P = NP$, להוכיח טענה זה קל באותה מידה כמו לבדוק בצורה טכנית שההוכחה נכונה.
הבעיה המשלימה היא: בהינתן P טענה מתמטית, לא קיימת הוכחה באורך t לטענה. זו כבר בעיה $coNP$־שלמה. לא ברור מהו העד לכך שאין הוכחה (שאפשר לבדוק בזמן פולינומי!), בעוד שהיה ברור לגמרי מהו העד לקיום ונכונות הוכחה - ההוכחה עצמה. אותו קושי נכון להרבה בעיות $coNP$־שלמות: איך משתכנעים שאין בגרף מעגל המילטון? איך משתכנעים שאין בגרף קליקה בגודל k?
האלטרנטיבה למוודאים שקיימת עבור $coNP$ היא "לדבר עם מישהו" בזמן פולינומי ולהשתכנע. קוראים לזה פרוטוקול אינטראקטיבי ונראה בהמשך ש־ $coNP \subseteq IP$.

6.9 אינטראקטאביליות (Intractability) וסימולציה בזמן

מטרתנו עבור $t(n)$ כלשהי למצוא $L \in \text{TIME}[t(n)]$ כך שעבור $r(n) < t(n)$, $L \notin \text{TIME}[r(n)]$. לדוגמא, נרצה למצוא $L \in \text{TIME}[n^3]$ אבל $L \notin \text{TIME}[n^2]$. מהדיון ינבע גם ש־ $P \subsetneq \text{EXPTIME}$, כלומר איפשהו בין P ל־ EXPTIME באמת מתווספות שפות חדשות.
<u>הרעיון הכללי</u>: יהיו לנו מכונות טיורינג M, M' כך ש־ M' מסיימת לחשב מהר יחסית ו־ M עובדת יותר זמן. רוצים להשתמש ב־ M כדי לחשב שפה ש־ M' לא יכולה לחשב. כדי ש־ M' תמיד לא תצליח לחשב את מה ש־ M מחשבת, M יכולה לסמלץ ריצה של M' ולענות את ההיפך. כמובן, M לא מכירה את M' ולכן צריכה לסמלץ את כל המכונות שרצות בזמן של M', ולעשות את זה בזמן יעיל. (כשראינו מכונת טיורינג אוניברסאלית, עלות הסימולציה הייתה $\mathcal{O}(n^2)$, וזה לא יאפשר לנו להפריד בין מחלקות כמו $\text{TIME}[n^3]$ ו־ $\text{TIME}[n^2]$.)

הגדרה 6.36 תהי $t : \mathbb{N} \to \mathbb{N}$ ומתקיים $t(n) = \Omega(n \log n)$. נאמר ש־ t <u>חשיבה בזמן</u> אם קיימת מכונת טיורינג שבהינתן הקלט 1^n מחשבת את $t(n)$ בייצוג בינארי בזמן $\mathcal{O}(t(n))$.
הסיבה לחסם התחתון היא שאפילו רק למצוא את אורך הקלט שנתון בייצוג אונארי לוקח זמן $n \log n$.

<u>דוגמאות</u>:
1. $t(n) = n^2$ חשיבה בזמן - לוקח $n \log n$ זמן למצוא את n (קידום מונה באורך $\log n$ במשך n צעדים) ואז העלאה בריבוע לוקחת זמן שהוא פולינומי ב־ $\log n$.
2. $t(n) = n\lfloor \log^2 n \rfloor$ חשיבה בזמן.
3. $t(n) = 2^n$ חשיבה בזמן $\mathcal{O}(n)$, שכן נכתוב 1 ואז על כל ספרה של הקלט נוסיף 0 לפלט.

משפט 6.37 (משפט הסימולציה בזמן) קיימת מכונת טיורינג S שבהינתן $(\langle M \rangle, w, t)$ מחשבת את הקונפיגורציה של M בריצתה על w במשך t צעדים, ומתקיים: אם $n = |\langle M \rangle| + |w|$ ו־ $t \geq n$ אזי S מסיימת לרוץ בזמן $\mathcal{O}\left(t(|\langle M \rangle|^3 + \log t)\right)$.

הוכחה: קצת אינטואיציה - בביטוי המפחיד שאנו רוצים להוכיח, $|\langle M \rangle|$ מופיע כי אנו צריכים גישה לקידוד של M בכל צעד של הסימולציה, ו־ $\log t$ מופיע כי צריך לספור מ־ 0 עד t את צעדי הסימולציה. כעת נעבור להוכחה.
נשים לב שבהינתן מצב q של M ואות a מתוך א"ב העבודה שלה, ניתן לחשב את $\delta_M(q, a)$ בזמן $\mathcal{O}(|\langle M \rangle|^3)$. איפה הבעיות?
1. צריך להפריד את $\langle M \rangle$ מהסרט של M תוך כדי סימולציה (אנחנו צריכים מכונה בעלת סרט אחד).
2. צריך להחזיר את הראש בכל פעם לקידוד של M וכשהריצה נמצאת בשלבים מתקדמים, יכול להיות שזה כבר ייקח יותר מאשר $\mathcal{O}(|\langle M \rangle|^3)$ זמן.
3. צריך לעצור אחרי t צעדים, כלומר צריך לשמור את המונה שסופר צעדים. אם שומרים אותו בתחילת הסרט, שוב כדי לעדכנו צריך לחזור לתחילת הסרט וזה ייקח זמן שגדול מ־ $\log t$.

אילוסטרציה:

M's head

↓

$t\,\#\langle M\rangle\,\#$ $\quad$ [.....]

M's tape

כדי לפתור את הבעיות הנ"ל, נצטרך לאחסן את המידע שמשתמשים בו בסמוך למקום שבו צריך אותו, כלומר בסמוך לראש המסומלץ. את תאי הסרט של S נחלק לשלוש שכבות, כלומר כל תא יהיה שלשה:

סרט העבודה של M - בקידוד לא"ב של S
המצב הנוכחי q והקידוד של M, משמאלו
המונה שסופר את מספר הצעדים שבוצעו

בנוסף, נצטרך סימן מיוחד, למשל $\dot{\square}$ (נקודה מעל), כדי לציין את המקום שבו הראש של M נמצא על סרט העבודה. נשים לב שגם q וגם $\langle M\rangle$ יהיו כתובים, ככל הנראה, על מספר תאים, אבל בכל מקרה בשכבה השניה <u>מתחת למקום שבו נמצא הראש</u> של M. כנ"ל לגבי המונה.

למשל, כאשר סרט החישוב של M מכיל ab, הראש נמצא מעל a, והמצב הוא q, תמונת המצב של הסרט של S תהיה כזו:

									$\dot{a}$					b								
									$\langle M\rangle$					q								
									המונה													

ולאחר שהראש זז ימינה והחליף את a ב־ c, תמונת המצב של הסרט של S תהיה כזו:

									c					$\dot{b}$								
									$\langle M\rangle$					q								
														המונה								

כלומר, בכל צעד אנו כותבים את המונה, המצב, והקידוד של המכונה מתחת לתא שאליו זזנו, וצריך לחסר מהמונה. כל צעד חישוב של M ניתן לפיכך לסמלץ בזמן $\mathcal{O}(|\langle M\rangle|^3+\log t)$ ולכן מתקבל החסם הנדרש לגבי עלות כל הסימולציה, שנמשכת t צעדים. ■

הערה 6.38 הדרישה $t \geq n$ נועדה להבטיח שיצירת המצב ההתחלתי (שלוקחת זמן ליניארי ב־ n) לא תשפיע על ההתנהגות האסימפטוטית של הסימולציה.

מסקנה 6.39 אם $t : \mathbb{N} \to \mathbb{N}$ פונקציה חשיבה בזמן, אז קיימת מכונת סימולציה S' המקיימת:

1. על קלט באורך n, S' רצה בזמן $\mathcal{O}(t(n))$;
2. על קלט מהצורה $(\langle M\rangle, w)$, S' מחשבת את הקונפיגורציה של M בריצתה על הקלט $(\langle M\rangle, w)$ במשך $\left\lfloor \frac{t(n)}{|\langle M\rangle|^3+\log t(n)} \right\rfloor$ צעדים.

הוכחה: פעולת המכונה תהיה כך. ראשית, מ־ $(\langle M\rangle, w)$ נחשב את $t' = \left\lfloor \frac{t(n)}{|\langle M\rangle|^3+\log t(n)} \right\rfloor$. ניתן למכונה S מהמשפט לטפל בקלט $(\langle M\rangle, (\langle M\rangle, w), t')$ ונחזיר את מה ש־ S מחזירה. (יש גם תקינות קלט שלוקחת זמן ליניארי.)

נבדוק זמנים: לחשב את $|\langle M\rangle| + |w|$ לוקח זמן $n \log n$ וזה בסדר כי t חשיבה בזמן (כלומר, חסומה מלרע ע"י $n \log n$). כדי לחשב את t', צריך לחשב את $t(n)$ - בזמן $\mathcal{O}(t(n))$ כי היא חשיבה בזמן, לחשב $\log t(n)$ לוקח זמן $\mathcal{O}(\log t(n))$, לחשב את $|\langle M\rangle|^3$ אפשר בזמן פולי־לוגריתמי, ולבסוף לחשב את המנה (בשלמים!) כנ"ל.

כלומר, חישוב t' לוקח זמן $\mathcal{O}(t(n))$. בניית הקלט ל־ S' זניחה לעומת הריצה של S שהוא $\mathcal{O}\left(t'(|\langle M\rangle|^3 + \log t')\right)$. גם כאן מקבלים $\mathcal{O}(t(n))$ כנדרש. ■

משפט 6.40 (משפט הסימולציה בזמן) אם $t : \mathbb{N} \to \mathbb{N}$ חשיבה בזמן, אזי קיימת שפה $L \in \mathrm{TIME}[t(n)]$ אבל אי אפשר להכריע את L בזמן $o\left(\frac{t(n)}{\log t(n)}\right)$.

הוכחה: נצביע במפורש על השפה הזאת. זוהי השפה הבאה:

$$L = \{(\langle M\rangle, w) : \; n := |\langle M\rangle| + |w|,\; t' := \left\lfloor \frac{t(n)}{|\langle M\rangle|^3 + \log t(n)} \right\rfloor : \\ M \underline{\text{ does not accept }} (\langle M\rangle, w) \text{ within } t' \text{ steps}\}$$

ראשית, ברור ש־ $L \in \mathrm{TIME}[f(n)]$ כי בהינתן $\langle M\rangle, w$ נריץ את M על w ונבדוק האם היא מקבלת תוך t' צעדים. לפי משפט הסימולציה, יש לנו מכונה שעושה את זה בזמן $\mathcal{O}(f(n))$, ואם M לא קיבלה - נקבל.

כעת נניח בשלילה שקיימת מ"ט M_0 המכריעה את L ורצה בזמן $r(n) = o\left(\frac{t(n)}{\log t(n)}\right)$. בפרט הדבר נכון כאשר M_0 רצה על קלט מהצורה $(\langle M_0\rangle, w)$. עבור n גדול מספיק מתקיים $r(n) < \left\lfloor \frac{t(n)}{|\langle M_0\rangle|^3 + \log t(n)} \right\rfloor = t'$ כיוון ש־ $|\langle M_0\rangle|$ קבוע. נבחר קלט מהצורה $(\langle M_0\rangle, w)$ שאורכו n מספיק גדול כנ"ל.

כעת אם M_0 מקבלת את $(\langle M_0\rangle, w)$ אז היא מקבלת אותו בזמן $r(n) < t'$ ולכן $(\langle M_0\rangle, w) \notin L$ וזו סתירה כי M_0 מכריעה את L. להיפך, אם M_0 דוחה את $(\langle M_0\rangle, w)$ אז זה אומר שהיא לא מקבלת תוך $r(n) < t'$ צעדים ולכן $(\langle M_0\rangle, w) \in L$ וזו שוב סתירה כי M_0 אמורה לקבל את $(\langle M_0\rangle, w)$ כי היא מכריעה את L. ■

7 סיבוכיות זיכרון

בפרק זה נתבונן על מדד אחר של "ביצועי" מכונת טיורינג - כמות הזיכרון שהיא משתמשת בה. נשים לב שלכל המודלים של מ"ט שראינו (מספר סרטים, RAM וכו') הצלחנו לעשות סימולציה בזיכרון ליניארי אפילו אם הזמן היה ריבועי או יותר. זה די מתבקש, שהרי אפשר להשתמש שוב ושוב באותו הזיכרון, מה שאי אפשר לעשות לגבי זמן.

7.1 מחלקות זיכרון, PSPACE ו־ NPSPACE

הגדרה 7.1 אם M מכונת טיורינג דטרמיניסטית שלכל קלט x עוצרת תוך שימוש ב־ $\mathcal{O}(f(|x|))$ תאי סרט, אז נאמר ש־ $L(M) \in \text{SPACE}[f(n)]$.

(המטרה של ה־ $\mathcal{O}$ כאן היא להבטיח שלא תהיה חשיבות לא"ב של הסרט, כל עוד הוא לא אונארי.)

אם N מכונת טיורינג לא דטרמיניסטית שלכל קלט x <u>עוצרת</u> תוך שימוש ב־ $\mathcal{O}(f(|x|))$ תאי סרט, ובנוסף:

1. אם N מקבלת את x, אז קיימת ריצה מקבלת.
2. אם N דוחה את x, אז כל הריצות דוחות.

אז נאמר ש־ $L(N) \in \text{NSPACE}[f(n)]$.

לגבי שפה כלשהי L, נאמר ש־ $L \in \text{SPACE}[f(n)]$ אם קיימת מ"ט דטרמיניסטית שמכריעה אותה בזמן $\mathcal{O}(f(n))$, וכנ"ל לגבי $\text{NSPACE}[f(n)]$.

הערה 7.2 יש בעיה עם ההגדרה הזאת עבור מחלקות מקום תת־ליניאריות, שכן הקלט עצמו כבר תופס מקום שהוא ליניארי באורך הקלט. כדי להתמודד עם זה, כל המכונות שלנו יהיו בעלות שני סרטים:

1. <u>סרט הקלט</u> - הקלט כתוב עליו ואסור לשנותו.
2. <u>סרט החישוב</u> - מותר לשנותו.

והשאלה על SPACE היא לגבי סרט החישוב. כעת f יכולה להיות גם פונקציה תת־ליניארית, כמו למשל log. למשל, כל שפה רגולרית תהיה ב־ $\text{SPACE}[0]$, כי קיימת מ"ט שמכריעה אותה והיא פשוט DFA.

הגדרה 7.3

$$\begin{aligned}
\text{PSPACE} &= \bigcup_{k\in\mathbb{N}} \text{SPACE}[n^k] \\
\text{NPSPACE} &= \bigcup_{k\in\mathbb{N}} \text{NSPACE}[n^k] \\
L = \text{LOGSPACE} &= \text{SPACE}[\log n] \\
NL = \text{NLOGSPACE} &= \text{NSPACE}[\log n]
\end{aligned}$$

לדוגמא, $\text{SAT} \in \text{PSPACE}$ - נעבור על כל ההשמות, כלומר נכתוב אחת על הסרט בכל רגע ונבדוק האם היא מספקת את הנוסחא. ברור שכמות הזיכרון היא פולינומיאלית - ולמעשה ליניארית - באורך הנוסחא.

הערה 7.4 בהינתן מכונה שעובדת ב־ $\text{TIME}[f(n)]$ ברור שהיא גם ב־ $\text{SPACE}[f(n)]$ שהרי היא לא יכולה להשתמש ביותר תאי סרט מאשר צעדי חישוב. לכן:

$$\begin{aligned}
\text{TIME}[f(n)] &\subseteq \text{SPACE}[f(n)] \\
\text{NTIME}[f(n)] &\subseteq \text{NSPACE}[f(n)]
\end{aligned}$$

להיפך, בהינתן מכונה שעובדת ב־ $\text{SPACE}[f(n)]$, יש מספר חסום של קונפיגורציות שמשתמשות ב־ $\mathcal{O}(f(n))$ תאי סרט. אם רצים יותר זמן, חוזרים על אותה קונפיגורציה פעמיים - ולכן הריצה לא תסתיים. קונפיגורציה מורכבת ממצב המכונה, מקום הראש

על הסרט הראשון, מקום הראש על הסרט השני, ומצב הסרט השני (שהרי מצב סרט הקלט תמיד קבוע). יש לנו $|Q|$ מצבים, $|x|$ אפשרויות למקום הראש על סרט הקלט, $\mathcal{O}(f(x))$ מצבים לראש על סרט החישוב, ו־ $|\Gamma|^{\mathcal{O}(f(|x|))}$ מצבים של סרט החישוב. זה חסם לא מדויק במיוחד, והוא נותן:

$$\mathrm{SPACE}[f(n)] \subseteq \mathrm{TIME}[n \cdot 2^{\mathcal{O}(f(n))}]$$

כאשר נשים לב שבסיס החזקה לא משנה בגלל ה־ $\mathcal{O}$ במעריך. באותו אופן,

$$\mathrm{NSPACE}[f(n)] \subseteq \mathrm{NTIME}[n \cdot 2^{\mathcal{O}(f(n))}]$$

נוכל להסיק מכאן:

$$P \subseteq \mathrm{PSPACE}, \quad NP \subseteq \mathrm{NPSPACE}, \quad L \subseteq P, \quad NL \subseteq NP$$

<u>שאלה פתוחה</u>: האם $L \subsetneq NL$?
מכאן נובע העניין בבעיות NL־שלמות וקשות, וכדי לדבר עליהן נצטרך להגדיר רדוקציה חלשה יותר מאשר רדוקציה פולינומיאלית. זאת משום שנראה בהמשך ש־ $NL \subseteq P$ ולכן רדוקציה פולינומיאלית היא חזקה מדי - הרדוקציה יכולה פשוט להכריע את הבעיה ולהמשיך הלאה. אנחנו נצטרך רדוקציה שרצה ב־ LOGSPACE.

7.2 שלמות ב־ NL, רדוקציית LOGSPACE

הבעיה הקנונית של NL היא PATH.

הגדרה 7.5

$$\mathrm{PATH} = \{\langle G, s, t\rangle : \ G \text{ is a directed graph, there is a path in } G \text{ from } s \text{ to } t\}$$

טענה 7.6 $\mathrm{PATH} \in NL$.

הוכחה: נציג מכונת טיורינג לא דטרמיניסטית שמכריעה את PATH ופועלת בזיכרון לוגריתמי באורך הקלט. האלגוריתם שהמכונה מממשת:
1. אתחל $v \leftarrow s$
2. באופן לא דטרמיניסטי, $v \leftarrow u : \ (v,u) \in E$ (מתוך כל הצלעות המקיימות זאת)
3. אם $v = t$, קבל
4. אחרת, חזור על שלב 2, אבל לא יותר מאשר $|V|$ פעמים
5. דחה

ברור שהמגבלה על $|V|$ תקינה כי אם יש מסלול, יש מסלול שמשתמש בלא יותר מאשר $|V|$ צלעות, בפרט מסלול פשוט, ונכונות האלגוריתם ברורה. כמות המקום שמשתמשים בו: אנו צריכים מקום למונה ול־ v, כל אחד מהם בין 1 ל־ $|V|$ ולכן כמות המקום - בייצוג בבסיס 2 - היא $\mathcal{O}(\log|V|)$ ולכן $\mathrm{PATH} \in NL$. ■

הגדרה 7.7 יהיו A, B שפות כלשהן. רדוקציית LOGSPACE זוהי פונקציה חשיבה $f : \Sigma_A^* \to \Sigma_B^*$ הפועלת ב־ LOGSPACE כך ש־ $x \in A \iff f(x) \in B$. המשמעות של "חשיבה ב־ LOGSPACE" היא שיש שלושה סרטים:
1. סרט הקלט - לקריאה בלבד.
2. סרט חישוב - לקריאה ולכתיבה.
3. סרט הפלט - לכתיבה בלבד.

והדרישה היא שמספר התאים שמשתמשים בהם בסרט החישוב יהיה $\mathcal{O}(\log n)$.
אם קיימת רדוקציית LOGSPACE בין A ל־ B, נסמן $A \leq_L B$.

טענה 7.8 אם $A \in NL$ אז $A \leq_L \mathrm{PATH}$.

הוכחה: בהינתן קלט x נבנה $\langle G, s, t\rangle$ כדלקמן: יש לנו את N, מ"ט לא דטרמיניסטית שמכריעה את A ב־ LOGSPACE. בגרף G, הקודקודים V יהיו הקונפיגורציות של N ו־ E הקשתות שמחברות בין שתי קונפיגורציות שניתן לעבור ביניהן באמצעות מעבר δ.
אנחנו לא צריכים את כל הקונפיגורציות, אלא רק קונפיגורציות עם סרט באורך $\mathcal{O}(\log|x|)$ - והקבוע של ה־ $\mathcal{O}$ ידוע לנו מהמכונה. יש מספר פולינומיאלי ב־ $|x|$ של קונפיגורציות כאלה. s יהיה הקונפיגורציה ההתחלתית על x, ו־ t תהיה קונפיגורציה מיוחדת שיש אליה מעבר מכל קונפיגורציה מקבלת (כך אנו מטפלים באפשרות שיש כמה קונפיגורציות מקבלות - למשל, t מייצגת את q_{acc} כאשר על הסרט לא כתוב שום דבר).
צריך להראות שהרדוקציה נכונה ופועלת ב־ LOGSPACE. כל קונפיגורציה מכילה $\mathcal{O}(\log|x|)$ ביטים - אינדקס, מספר המצב, מצב הסרט שהוא בעצמו בגודל $\mathcal{O}(\log|x|)$, ומצב הראש שהוא מספר בין 1 ל־ $|x|$; לאחר מכן עלינו לעבור על זוגות קונפיגורציות (עדיין LOGSPACE) וכמובן כנ"ל לגבי s, t. הפלט כאן הוא בכלל לא LOGSPACE, אבל על סרט החישוב אנחנו בהחלט משתמשים רק ב־ $\mathcal{O}(\log|x|)$ מקום. כמובן, הנכונות קלה.
לכן PATH היא NL־שלמה. ■

הערה 7.9 יש אלגוריתם פולינומיאלי בזמן שמכריע את PATH, למשל BFS או Dijkstra. לכן $\text{PATH} \in P$. כמובן, אם יש רדוקציה חשיבה ב־ LOGSPACE אז מהאבחנות שלנו קודם היא חשיבה ב־ P כמות שהיא (מספר הקונפיגורציות הוא פולינומיאלי), כלומר $\leq_L$ גורר $\leq_p$. מזה נסיק ש־ $NL \subseteq P$.
היררכיית המחלקות הידועה לנו כרגע:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq \text{NPSPACE}$$

וגם:

$$P \subseteq \text{PSPACE} \subseteq \text{NPSPACE}$$

טענה 7.10 אם $A \leq_L B$ ו־ $B \in NL$ אז $A \in NL$.

הוכחה: נשים לב שזה לא טריוויאלי - הפלט של הרדוקציה יכול להיות פולינומיאלי, אין לנו חסם עליו בניגוד לרדוקציה פולינומיאלית בזמן. הרעיון הכללי הוא כזה: לא חייבים להחזיק את כל $f(x)$ אלא אפשר לחשב אותו מספר פעמים.
בהינתן x קלט של A ורדוקציית LOGSPACE ל־ B שנסמנה f, אנחנו מסמלצים את המכונה M שמכריעה את B. בכל פעם ש־ M מנסה לקרוא קלט, אנחנו נחשב את $f(x)$ ב־ LOGSPACE ונפלוט רק את האות שהראש של M צריך כרגע - את זה נעשה לכל צעד של M. כיוון ש־ B פועלת ב־ LOGSPACE ואנחנו לא שומרים את כל $f(x)$ אף פעם, מקבלים מכונה להכרעת A ב־ LOGSPACE כנדרש. ■

הגדרה 7.11 מוודא LOGSPACE זוהי מכונת טיורינג בעלת שלושה סרטים שהקלט שלה הוא זוג x, y. הסרט הראשון הוא סרט קלט שעליו כתוב x, וזהו סרט לקריאה בלבד. הסרט השני הוא סרט העד שעליו כתוב y, וזהו סרט לקריאה בלבד בכיוון אחד (ימינה בלבד). הסרט השלישי הוא סרט העבודה, ואנו דורשים שצריכת המקום עליו תהיה $\mathcal{O}(\log|x|)$.

טענה 7.12 $A \in NL$ אסם קיים מוודא LOGSPACE עבורה V כך ש־ $A = \{x : \exists y,\ \langle x, y\rangle \in L(V)\}$.

הוכחה: (תרגיל 12, שאלה 3) סקיצת הוכחה: בהינתן מוודא, מ"ט לא דטרמיניסטית עבור A יכולה לסמלץ את המוודא על סרט העבודה שלה, כאשר סרט הקלט שלה הוא גם סרט הקלט של המוודא. מה לגבי סרט העד? ובכן, בכל פעם שהמוודא מבקש אות מסרט העד, המכונה שלנו בוחרת אות באופן לא דטרמיניסטי - אין צורך לשמור את האות כי הסרט הוא חד־פעמי.
להיפך, העד הוא מהלך הריצה המקבלת ("כתובת") של המ"ט הלא דטרמיניסטית עבור A על הקלט. המוודא יכול לבדוק את תקינות המהלך והעובדה שהריצה מקבלת בדומה להוכחה עבור מוודאים ב־ NP. ■

7.3 שלמות ב־ PSPACE

אנו רוצים למצוא שפה שלמה ב־ PSPACE, ומשתמשים ברדוקציות <u>פולינומיאליות בזמן</u> לשם כך. אפשר היה באופן עקרוני להגדיר רדוקציית PSPACE ולהשתמש בה, אבל המטרה שלנו היא להפריד בין P ל־ PSPACE. כיוון שבמילא $P \subseteq \text{PSPACE}$, רדוקציית PSPACE מבעיה ב־ P יכולה בעצמה לפתור את הבעיה, כלומר היא חזקה מדי.
השפה שנתבונן בה תהיה TQBF, שפת הנוסחאות הבוליאניות הסגורות (עם כמתים - Totally Quantified Boolean Formula) שערך האמת שלהן הוא $\mathbb{T}$.
בנוסחא כזו אנו מרשים להשתמש גם בכמת $\exists$ וגם בכמת $\forall$, ודורשים שכל המשתנים יהיו תחת כמת. לכן, ערך האמת של הנוסחא קבוע ואינו תלוי בהשמה על המשתנים. למשל, לנוסחא $\forall x \exists y (x \vee y) \wedge (\overline{x} \vee \overline{y})$ יש ערך אמת $\mathbb{T}$, שכן לכל x אפשר לבחור $y = \overline{x}$. לעומת זאת, לנוסחא $\exists y \forall x (x \vee y) \wedge (\overline{x} \vee \overline{y})$ יש ערך אמת $\mathbb{F}$. נרשה להכניס לנוסחא גם את הקבועים $\mathbb{T}, \mathbb{F}$.

טענה 7.13 $\text{TQBF} \in \text{PSPACE}$.

הוכחה: אלגוריתם להכרעת TQBF במקום פולינומיאלי על הנוסחא φ:
1. אם ב־ φ יש רק קבועים, נעבור עליה ונחשב את ערך האמת.
2. אחרת, אם הכמת הראשון הוא $\exists$, כלומר $\varphi = (\exists x)\varphi'$, אז נציב פעם אחת $x = \mathbb{T}$ ופעם אחת $x = \mathbb{F}$, ובכל פעם נבצע קריאה רקורסיבית כדי לבדוק האם הנוסחא החדשה היא ב־ TQBF. נחזיר "כן" אם אחת ההצבות הצליחה.
3. אם $\varphi = (\forall x)\varphi'$, נעשה כנ"ל אבל נחזיר "כן" רק אם שתֵי ההצבות הצליחו.

קל להוכיח את נכונות האלגוריתם באינדוקציה על מספר הכמתים בנוסחא. לגבי כמות המקום שמשתמשים בה: יש כאן רקורסיה, ובכל שלב של הרקורסיה אנחנו צריכים לזכור מה היה הערך - כלומר צריך מחסנית קריאות. כל frame במחסנית תופס רק מקום קבוע (כי אנחנו לא צריכים לשמור את כל הנוסחא, וגם אם כן זה יהיה בסדר), ועומק הרקורסיה הוא כמספר המשתנים. לכן כמות המקום היא אכן פולינומיאלית באורך הנוסחא. ■

הערה 7.14 בכל הדיון הזה לא הוכחנו ש־ TQBF היא שלמה ב־ PSPACE. ניתן למצוא את ההוכחה המלאה בוויקיפדיה או בספר, והיא מתבססת על הרעיון של CANYIELD מהמשפט הבא.

עיקר העניין הוא שאפשר לבנות נוסחא המייצגת את מצב המכונה (בדומה למה שעשינו בהוכחה ש־ SAT היא NP־שלמה), ואף נוסחא עבור קונפיגורציות $\varphi_{c_1,c_2,t}$ שמקבלת ערך אמת $\mathbb{T}$ אסם המכונה המתאימה יכולה לעבור מ־ c_1 ל־ c_2 תוך t צעדים. אז, השאלה על קבלת מילה תוך x צעדים מצמטצמת לבדיקת הנכונות של הנוסחא $\varphi_{c_0,c_{acc},x}$.

כאן העסק מסתבך מעט כי צריך להראות שאפשר לחשב את הרדוקציה בזמן פולינומי, ובאופן עקרוני עושה רושם שהנוסחא מתפוצצת באופן אקספוננציאלי. אבל, אנו יכולים לנצל את שני הכמתים כדי לשים לב למאפיין הבא:

$$\varphi_{c_1,c_2,t} \equiv \exists c' \forall c_3, c_4 \in \{(c_1, c'), (c', c_2)\} \varphi_{c_3,c_4,\lceil \frac{t}{2} \rceil}$$

ואת הנוסחא הזאת אפשר לבנות בזמן פולינומיאלי.

משפט 7.15 (משפט Savitch) אם $f(n) = \Omega(\log n)$ אז:

$$\text{NSPACE}[f(n)] \subseteq \text{SPACE}[f^2(n)]$$

הוכחה: אנו נעשה כמה הנחות מקלות שמגבילות את הכלליות - נדרוש $f(n) = \Omega(n)$ וגם ש־ f חשיבה ב־ $\text{SPACE}[f(n)]$. נניח ש־ N מ"ט לא דטרמיניסטית שעובדת ב־ $\text{SPACE}[f(n)]$ ונבנה מ"ט דטרמיניסטית שעובדת ב־ $\text{SPACE}[f^2(n)]$.

נרצה לשאול על N את השאלה $\text{CANYIELD}(c_1, c_2, t)$ שמשמעותה - האם ניתן להגיע מהקונפיגורציה c_1 לקונפיגורציה c_2 תוך ביצוע t צעדי חישוב.

להלן האלגוריתם $\text{CANYIELD}(c_1, c_2, t)$:
1. אם $t = 1$, ניתן לענות מהתבוננות ב־ δ_N.
2. אחרת, לכל c' אפשרית: אם גם $\text{CANYIELD}(c_1, c', \lfloor \frac{t}{2} \rfloor)$ וגם $\text{CANYIELD}(c', c_2, \lceil \frac{t}{2} \rceil)$ קבל.
3. דחה.

נכונות האלגוריתם ברורה מאינדוקציה על t. צריך לציין שאנו מסתכלים על c' שאורך תוכן הסרט שלה הוא לא יותר מאשר $\mathcal{O}(f(n))$, מהנתון על N.

צריכת המקום של האלגוריתם: כל רמה של הרקורסיה דורשת לשמור את c' כדי שנוכל למצוא את ה־ c' הבא, וכן עוד מספר קבוע של קונפיגורציות. כלומר, כל רמה צורכת מקום $\mathcal{O}(f(n))$ וכן צריך לשמור את t. עומק הרקורסיה הוא $\lceil \log_2 t \rceil$.

כעת בהינתן x נרצה לבדוק האם N מקבלת את x, וזה קורה אסם יש ריצה מהקונפיגורציה ההתחלתית על x לקונפיגורציה מקבלת כלשהי. נדרוש שתהיה ל־ N קונפיגורציה מקבלת יחידה - צריך בשביל זה מעבר מ־ q_{acc} למצב מקבל חדש שבדרך אליו מוחקים את כל הסרט ושמים את הראש במקום הראשון - וזה כמובן לא מגביל את הכלליות.

מספר הקונפיגורציות האפשריות בסה"כ הוא $2^{\mathcal{O}(f(n))}$, כלומר N תרוץ לכל היותר במשך $2^{\mathcal{O}(f(n))}$ צעדים. לכן, N מקבלת את x אסם $\text{CANYIELD}(c_0, c_{acc}, 2^{\mathcal{O}(f(n))})$ כאשר c_0, c_{acc} הקונפיגורציות ההתחלתית והמקבלת בהתאמה.

כיוון שאנו יודעים את t, עומק הרקורסיה הוא $\mathcal{O}(f(n))$ עם צריכת זיכרון של $\mathcal{O}(f(n))$ לכל frame, ולכן סה"כ צריכת הזיכרון היא $\mathcal{O}(f^2(n))$ כנדרש. נשים לב שכדי לרשום את t אנו צריכים $\log(2^{\mathcal{O}(f(n))})$ מקום, אבל זה בדיוק $\mathcal{O}(f(n))$ עדיין. ■

מסקנה 7.16 $\text{NPSPACE} = \text{PSPACE} = \text{coPSPACE} = \text{coNPSPACE}$

הוכחה: מסקנה ישירה מהמשפט והעובדה ש־ $\text{PSPACE} = \text{coPSPACE}$ שכן זו מחלקה של שפות המוגדרות ע"י מכונות דטרמיניסטיות ואפשר להחליף את q_{acc} ו־ q_{rej} כפי שראינו בעבר כדי לקבל את השפה המשלימה. ■

טענה 7.17 תהי $f : \mathbb{N} \to \mathbb{N}$ כך ש־ $f(n) \geq n$. אז מתקיים $\text{NTIME}[f(n)] \subseteq \text{SPACE}[f(n)]$.

הוכחה: (תרגיל 11, שאלה 5) סקיצת הוכחה: בהינתן מ"ט לא דטרמיניסטית N המכריעה $L \in \text{NTIME}[f(n)]$, בונים מ"ט דטרמיניסטית הפועלת במקום $\mathcal{O}(f(n))$ שמבצעת סימולציה של כל הריצות האפשריות של N. כדי לכתוב את הריצה הנוכחית אנו צריכים $\mathcal{O}(f(n))$ מקום וכדי לבצע את הסימולציה עצמה - כנ"ל. ■

טענה 7.18 השפה $A_{LBA} = \{\langle M, w\rangle : \ M \text{ is an LBA that accepts } w\}$ היא PSPACE-שלמה.

הוכחה: (תרגיל 12, שאלה 2) סקיצת הוכחה: ניזכר ש־ A_{LBA} זוהי מכונת טיורינג דטרמיניסטית שאסור לה לחרוג בעבודתה מעבר לקטע הסרט שעליו נמצא הקלט שלה. ההוכחה היא באמצעות רדוקציה מכל $L \in \text{PSPACE}$. יש ל־ L מ"ט D הפועלת במקום $cn^k \geq$ על מילים באורך n. בהינתן w, הרדוקציה מייצרת את הזוג $\langle M, w\#^{c|w|^k}\rangle$ כאשר M פועלת כמו D כשהיא מתייחסת ל־ $\#$ כמו למקום ריק על הסרט. כמובן, המקום שיש ל־ M מספיק מההנחה על D. ■

7.4 סימולציה במקום

נרצה להגיע למשפט סימולציה במקום ומשפט היררכיה במקום בדומה למה שעשינו לגבי זמן.

הגדרה 7.19 המקיימת $s : \mathbb{N} \to \mathbb{N}$ $s(n) \geq \log n$ נקראת <u>חשיבה במקום</u> אם יש מ"ט M שבהינתן קלט מהצורה 1^n פולטת את הייצוג הבינארי של $s(n)$ תוך שימוש במקום $\mathcal{O}(s(n))$.

משפט 7.20 (משפט הסימולציה במקום) קיימת מ"ט S שבהינתן $(\langle M\rangle, w, s)$ מקיימת את התנאים הבאים:
1. אם בריצתה על w המכונה M משתמשת ביותר מ־ s תאי סרט, או נכנסת ללולאה אינסופית, אז S תדחה.
2. אם בריצתה על w המכונה M משתמשת ב־ $s \geq$ תאי סרט ועוצרת, אז S תענה כמו M.
3. S משתמשת לכל היותר ב־ $\mathcal{O}(s \log n)$ תאי סרט, כאשר $n = |\langle M\rangle|$.

הוכחה: (חלקית בלבד) אנו צריכים לסמלץ s תאים של M, כל אחד בקידוד שלא יעלה על $\log n$ אותיות בא"ב של S. כמו כן נשים לב שמכונה המשתמשת ב־ s תאי סרט ולא נתקעת רצה לכל היותר $\mathcal{O}(n^s)$ צעדים. לכן:
1. נחשב את n^s. אם הסימולציה תימשך יותר צעדים מזה, נדחה. (נשים לב שהמקום הדרוש לשמירת מספר זה הוא בדיוק $s \log n$.)
2. נקצה על סרט העבודה מקום ל־ s תוים של הא"ב־סרט של M. תוך כדי הסימולציה, אם חורגים מהמקום הנ"ל - נדחה.
3. נבצע את הסימולציה כאשר סופרים בכל פעם כמה צעדים כבר עשינו, ונחזיר לבסוף את מה ש־ M מחזירה. כאן למשל נצטרך לרשום את המצב שמסמלצים תוך שימוש בא"ב של כמה שכבות, כמו שעשינו במשפט הסימולציה בזמן. ■

משפט 7.21 (משפט ההיררכיה במקום) אם $s : \mathbb{N} \to \mathbb{N}$ חשיבה במקום, אז קיימת $L \in \text{SPACE}[s(n)]$ אבל L לא ניתנת להכרעה במקום $o(s(n))$.

הוכחה: (תרגיל 12) ההוכחה דומה מאוד להוכחת משפט ההיררכיה בזמן. אנו בונים את השפה

$$L = \{\langle M, w\rangle : \ M \text{ does not accept } \langle M, w\rangle \text{ using } \frac{f(|\langle M, w\rangle|)}{\log |\langle M\rangle|} \text{ space}\}$$

ראשית, $L \in \text{SPACE}[s(n)]$ כי אפשר לחשב את $k = \frac{s(|\langle M,w\rangle|)}{\log |\langle M\rangle|}$ במקום $\mathcal{O}(s(n))$, כי s חשיבה במקום. כעת מריצים את מכונת הסימולציה על $\langle M, w, k\rangle$ וזה נעשה תוך זמן $k \log |\langle M\rangle| = s(n)$ כנדרש.
כעת מניחים בשלילה שקיימת מ"ט T המכריעה את L במקום $o(s(n))$. אז תהי $r(n)$ פונקציית סיבוכיות המקום של מכונה כזאת, ועבור n גדול מספיק מתקיים $r(n) < \frac{s(n)}{\log |\langle T\rangle|}$. נתבונן בהתנהגות של T על מילת קלט כלשהי $\langle T, w\rangle$ שארוכה מ־ n זה. כמו קודם, אם T מקבלת את הקלט, היא עושה זאת בשימוש בפחות מ־ $\frac{s(|\langle T,w\rangle|)}{\log |\langle T\rangle|}$ מקום, ואז הקלט אינו בשפתה, וזו סתירה. וגם להיפך - אם T אינה מקבלת את הקלט, אז מילת הקלט <u>כן</u> בשפתה, וזו שוב סתירה. ■

מסקנה 7.22 $L \subsetneq \text{PSPACE}$

והנה המצב המעודכן ביותר של מחלקות בזמן והמקום הידועות לנו כרגע:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$$

אנו יודעים שאחת ההכלות בין P ל־ EXPTIME היא הכלה ממש בגלל משפט ההיררכיה בזמן, ויודעים שאחת ההכלות בין L ל־ PSPACE היא הכלה ממש בגלל משפט ההיררכיה במקום. לגבי שאר ההכלות, אנחנו רק מאמינים שהן הכלות ממש.

8 פרוטוקולים אינטראקטיביים ורנדומיות

בפרק זה נראה מודלים חישוביים חדשים, למשל מודל שבו "מוכיח" ו"מוודא" המנהלים ביניהם תקשורת פולינומיאלית כדי להשתכנע בנכונות של טענה כלשהי בהסתברות גבוהה.

8.1 מבוא לרנדומיות במכונות טיורינג

הגדרה 8.1 מכונת טיורינג המשתמשת ברנדומיות היא בעלת שלושה סרטים:

1. סרט הקלט - לקריאה בלבד.
2. סרט עבודה - לקריאה וכתיבה, כרגיל.
3. סרט אקראיות - מחרוזת בינארית של $0/1$ באורך מסוים שניתן לקרוא כל ביט שלה פעם אחת בלבד, כאשר כל סדרה בינארית מופיעה על הסרט בהסתברות אחידה. כל קריאה מסרט האקראיות שקולה להטלת מטבע.

המכונה פועלת על הקלט x ועל המחרוזת האקראית R. עבור x נתון, אנו יכולים לשאול עבור כמה מה־ R־ים האפשריים המכונה תקבל ועבור כמה היא תדחה, ונסמן:

$$\Pr[M(x) = \mathcal{ACC}] \qquad \Pr[M(x) = \mathcal{REJ}]$$

כך נוכל להגדיר מספר מודלים של חישוב הסתברותי:

1. RP - random polynomial time - אסור לטעות:

$$\begin{aligned} x \in L &\implies \Pr[M(x) = \mathcal{ACC}] \geq \frac{1}{2} \\ x \notin L &\implies \Pr[M(x) = \mathcal{REJ}] = 1 \end{aligned}$$

2. $coRP$:

$$\begin{aligned} x \in L &\implies \Pr[M(x) = \mathcal{ACC}] = 1 \\ x \notin L &\implies \Pr[M(x) = \mathcal{REJ}] \geq \frac{1}{2} \end{aligned}$$

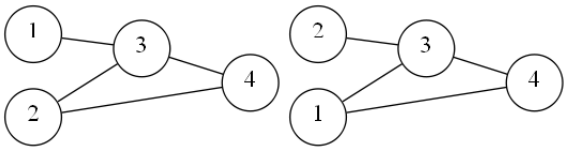
3. BPP:

$$\begin{aligned} x \in L &\implies \Pr[M(x) = \mathcal{ACC}] \geq \frac{2}{3} \\ x \notin L &\implies \Pr[M(x) = \mathcal{REJ}] \geq \frac{2}{3} \end{aligned}$$

הערה 8.2 אנו יודעים ש־ $P \subseteq RP \subseteq BPP$, ורוב החוקרים סבורים ש־ $BPP \subseteq P$.

8.2 מבוא לפרוטוקולים אינטראקטיביים

הגדרה 8.3 שני גרפים G_1, G_2 נקראים <u>איזומורפיים</u> אם הם זהים עד כדי שמות הקודקודים. לדוגמא, הגרפים הבאים הם איזומורפיים - אפשר לקבל את G_1 מ־ G_2 ע"י החלפת הקודקוד 2 ב־ 1.



הערה 8.4 כמובן, $\text{GraphIso} \in NP$ שכן העד זוהי פשוט פרמוטציה של הקודקודים, וכמובן הווידוא לוקח זמן פולינומיאלי. לעומת זאת, להשתכנע בכך ששני גרפים הם <u>לא</u> איזומורפיים זה די קשה, אלא אם כן למשל הדרגות של הקודקודים אינן אותו הדבר. אם זאת, מעצם ההגדרה, $\text{GraphNonIso} \in coNP$.

אנו זקוקים למודל של אינטראקציה - המכונה הפולינומיאלית "מדברת" עם גוף כל־יכול (נקרא לו "קהל") שיכול לענות על שאלות, שמחליף את ה"עד" מקודם. הפרוטוקול הוא כזה:

1. מטילים מטבע כדי לדעת האם מסתכלים על G_1 או על G_2.
2. מגרילים פרמוטציה אקראית π ומפעילים אותה על הגרף שנבחר.
3. מראים ל"קהל" את הגרפים ושואלים - מאיזה מהם הגענו לגרף החדש לאחר הפעלת π? (כלומר, מה הביט שהגרלנו בשלב 1.)
4. אם הקהל צדק, נאמר ש־ G_1 ו־ G_2 אינם איזומורפיים. (ולהיפך.)

הנקודה היא שאם הקהל הכל־יכול הזה טועה, אז הגרפים איזומורפיים - שכן אפשר לקבל את הגרף שהתקבל לאחר הפעלת π גם מ־ G_1 וגם מ־ G_2 (רק צריך עוד פרמוטציה בדרך). אם כן, במקרה שהגרפים לא איזומורפיים אנו תמיד צודקים. אם הגרפים כן איזומורפיים, אנו צודקים בהסתברות $\leq \frac{1}{2}$.

אמנם, ההסתברות הזאת ($\frac{1}{2}$) לא נראית טובה במיוחד - אבל אנחנו יכולים לחזור על הפרוטוקול שוב ושוב ובכל פעם אנו מקטינים את הסיכוי לטעות באופן אקספוננציאלי. אחרי k הפעלות של הפרוטוקול נקבל הסתברות לטעות של $\leq \frac{1}{2^k}$.

נשים לב גם שהפרוטוקול רץ בזמן פולינומיאלי.

הגדרה 8.5 מחלקת השפות שאפשר להכריע ע"י ביצוע פרוטוקול כנ"ל בזמן פולינומיאלי נקראת IP.

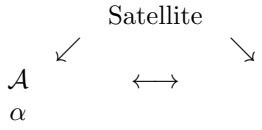
8.3 בעיית הזהות (Identity)

בעיית הזהות מוגדרת כך: שני אנשים, אליס ובוב, מקבלים שניהם מחרוזות בינאריות. אליס מקבלת את המחרוזת $x \in \{0,1\}^n$ ובוב מקבל את המחרוזת $y \in \{0,1\}^n$. בוב טוען ש־ $y = x$ ורוצה להוכיח זאת לאליס.

Alice		Bob
$\mathcal{A}$	$\longleftrightarrow$	$\mathcal{B}$
$x \in \{0,1\}^n$		$y \in \{0,1\}^n$

פתרון טריוויאלי הוא להעביר לאליס את כל y, ודרושים לשם כך n ביטים. במקום זאת, אנו רוצים פתרון הסתברותי (למשל, עם הסתברות לטעות $< \frac{1}{4}$) אבל תוך שימוש במספר קבוע של ביטים.

הרעיון הוא להשתמש במקור אקראיות משותף, "לווין בחלל" שיבחר ביטים רנדומיים וישדר אותם ליקום. גם אליס וגם בוב יוכלו להאזין לביטים האלה, ולדעת שהם קיבלו את אותם ביטים של רנדומיות. (מכאן - רנדומיות משותפת.)



לשם הפשטות, אנו נניח שבוב לא מנסה לרמות ולשכנע את אליס ש־ $\alpha = \beta$ כשהם לא. הדרישות האחרות שלנו מהפרוטוקול הן כאלה:

1. שלמות (completeness) - אם $\alpha = \beta$ אז $\Pr[\text{"yes"}] = 1$.
2. נאותות (soundness) - אם $\alpha \neq \beta$ אז $\Pr[\text{"yes"}] \leq \frac{1}{2}$.

כמו כן, רוצים לצמצם ככל האפשר את מספר הביטים המועברים.

הצעות:

1. נתייחס לביטים שהלווין משדר כאינדקס i בין 0 ל־ $n-1$, ונשלח את β_i לאליס. אליס יודעת שנשלח לה β_i (כי הרנדומיות משותפת) ומשווה $\alpha_i \stackrel{?}{=} \beta_i$. אם הם שונים, אליס תדחה; אם הם שווים, נמשיך לעשות את זה $\frac{n}{2}$ פעמים. ברור שהשלמות מתקיימת, וגם הנאותות מתקיימת כי הסיכוי שהביט השונה לא ייבחר הוא $\frac{1}{2}$. מספר הביטים המועברים הוא $\frac{n}{2}$ במקום n.

2. נריץ פונקציית hash כלשהי - משתמשים בביטים המקריים כדי לבחור פונקציה מסוימת מתוך משפחת פונקציות hash ומשווים את תוצאת הפעלת הפונקציה, שיכולה להכיל מספר קבוע של ביטים.

3. בעזרת הרנדומיות המשותפת אליס ובוב יבחרו איבר $z = (z_0, \ldots, z_n) \sim \mathcal{U}(\{0,1\}^n)$ - כלומר מחרוזת בינארית רנדומית באורך n בהתפלגות אחידה. כעת כל צד יחשב:

$$a = \left(\sum_{i=0}^{n-1} z_i \alpha_i\right) \bmod 2$$

$$b = \left(\sum_{i=0}^{n-1} z_i \beta_i\right) \bmod 2$$

בוב ישלח לאליס את b; אליס תקבל אסם $a = b$.
כמובן, מועבר רק ביט אחד ומתקיימת שלמות. הנאותות נובעת מהתרגיל הבא בהסתברות:
"מטילים 100 מטבעות הוגנים ובלתי תלויים. מה ההסתברות שמספר הפעמים שיופיע "פלי" הוא זוגי?"
מקרה א': עד המטבע ה־ 99 ראינו מספר זוגי של "פלי". הסיכוי שהמטבע האחרון ישאיר את זה כך הוא $\frac{1}{2}$.
מקרה ב': עד המטבע ה־ 99 ראינו מספר אי־זוגי של "פלי". גם כאן הסיכוי הוא $\frac{1}{2}$.
בעצם, לא היינו צריכים את ההוגנות/אי־תלות של 100 המטבעות; די שאחד מהם הוא הוגן ולא תלוי בשאר.
לענייננו, $a = b$ אסם $\sum z_i \alpha_i = \sum z_i \beta_i$ אסם $(\sum z_i(\alpha_i - \beta_i)) \bmod 2 = 0$. נשים לב שאם $\alpha \neq \beta$ אז יש i כך ש־ $\alpha_i \neq \beta_i$ ולכן קיבלנו את אותה הבעיה כמו עם המטבעות - אנו רוצים לדעת את הזוגיות של מספר ה־ z_i־ים שהם 1 עבור ה־ i־ים שבהם $\alpha_i \neq \beta_i$:

$$\sum_{i=0}^{n-1} z_i(\alpha_i - \beta_i) \equiv_{\mathrm{mod}2} \sum_{i:\alpha_i \neq \beta_i} z_i$$

ולכן הסכום הזה הוא 0 (מודולו 2) בהסתברות $\frac{1}{2}$ בדיוק כמו שרצינו עבור הנאותות.

4. מתבקש עידון כדי לקבל נאותות "טובה יותר", למשל הסתברות $\frac{1}{4}$ לטעות. כדי לקבל זאת, אפשר לבצע את הפרוטוקול פעמיים - כלומר בוחרים שני z־ים ומעבירים שני ביטים. אפשר להקטין את ההסתברות לטעות כרצוננו ל־ $\frac{1}{2^k}$ ע"י העברת k ביטים כאלה.

5. וריאציה על הבעיה: מה קורה אם בוב מנסה לשכנע את אליס ש־ $\alpha = \beta$ למרות שזה לא באמת המצב? (שאלה למחשבה - התשובה היא שזה לא עוזר לו.)

6. וריאציה נוספת על הבעיה: מה קורה אם אין רנדומיות משותפת, אלא לכל צד יש רנדומיות פרטית? הפרוטוקול הנ"ל, כמובן, לא עובד יותר, וגם כל מני רעיונות פשוטים אחרים לא עובדים.

8.4 פולינומים

טענה 8.6 אם $p(x)$ פולינום שאינו פולינום האפס ו־ $\deg(p) = d$, אז מספר השורשים של p הוא $\geq d$.

מסקנה 8.7 אם $p(x) \neq q(x)$ ושניהם מדרגה $\geq d$, אז $|\{z : p(z) = q(z)\}| \leq d$.

מכאן מתקבל הפרוטוקול לבעיית הזהות תוך שימוש ברנדומיות פרטית:

1. אליס תגדיר פולינום $p(x) = \sum_{i=0}^{n-1} \alpha_i x^i$
2. בוב יגדיר פולינום $q(x) = \sum_{i=0}^{n-1} \beta_i x^i$
3. אליס תגריל $z \sim \mathcal{U}(\{1, \ldots, n^2\})$ ותשלח אותו לבוב
4. בוב ישלח את $q(z)$ לאליס
5. אליס מקבלת אסם $p(z) = q(z)$

כעת נשים לב שאם $\alpha = \beta$ אז $p \equiv q$ ולכן $\Pr[\text{"yes"}] = 1$. לעומת זאת, אם $\alpha \neq \beta$ אז $p(x) \neq q(x)$ ומתקיים $\deg(p), \deg(q) \leq n - 1$. מכאן

$$\Pr[\text{"yes"}] = \Pr[p(z) = q(z)] \leq \frac{n-1}{n^2} \leq \frac{1}{n}$$

שהרי מתוך n^2 הערכים שמתוכם z מוגרל, יש לכל היותר $n - 1$ ערכים שעליהם $p(z) = q(z)$.
כמה ביטים מועברים כאן? כאשר השדה הוא $\mathbb{Q}$, מספר הביטים הדרושים להעברת z יכול להיות מאוד גדול. כדי לטפל בזה נעבוד מעל שדה סופי $\mathbb{Z}_p$ כש־ p ראשוני לא גדול במיוחד.
כדי לתקן את הפרוטוקול, אליס ובוב יבחרו את p להיות הראשוני הראשון שגדול מ־ n^2; מספיק לנו לדעת ש־ $n^2 < p \leq 2n^2$. מכאן כל הפרוטוקול נשאר אותו הדבר, הנאותות והשלמות נשמרות, אבל כמות הביטים שעוברים היא $\mathcal{O}(\log n)$, שהרי ב־ z יש לכל היותר $2 \log n$ ביטים וב־ $q(z)$ יש לכל היותר $2 \log p \leq 4 \log n$ ביטים.

הגדרה 8.8 פולינום בכמה משתנים הוא סכום של מונומים, כאשר כל מונום הוא מהצורה

$$a_{(e_i)_{i=1}^n} x_1^{e_1} \cdots x_n^{e_n}$$

כאשר כל $e_i \geq 0$ טבעי, האינדקס של המקדם a הוא סדרת החזקות, והדרגה של המונום היא $\sum_{i=1}^{n} e_i$. ניתן להציג את הפולינום בתור

$$f(x_1, \ldots, x_n) = \sum_{(e_i)} a_{(e_i)} \prod x_j^{e_j}$$

ללא המקדמים, יש לנו $\binom{n+d-1}{n-1}$ מונומים שונים מדרגה d. זה אקספוננציאלי, כלומר כדי לתאר פולינום מדרגה d מעל שדה $\mathbb{Z}_p$ אנו נצטרך מקום אקספוננציאלי. אם נבחר בייצוג שבו כותבים רק את המונומים שמשתתפים בסכום, אפשר לכתוב את חלק מהפולינומים האלה מעל $\mathbb{Z}_p$ במקום פולינומיאלי - בתנאי שמשתתף בהם מספר פולינומיאלי של מונומים.

אנו נשתמש בייצוג של פולינומים כאלה באמצעות נוסחא אריתמטית.

הגדרה 8.9 נוסחא אריתמטית מוגדרת באינדוקציה ע"י:

1. משתנה x_i או קבוע 1 הוא נוסחא.
2. אם p, q נוסחאות אז $p+q,\ p-q,\ p \cdot q,\ (-q)$ גם כן נוסחאות.

למשל, את הפולינום $x^3y + y + 2$ נוכל לייצג בתור $x \cdot x \cdot x \cdot y + y + 1 + 1$, ונוכל גם להוסיף סוגריים לפי הצורך.

טענה 8.10 אם F נוסחא באורך n, היא מגדירה פולינום שדרגתו לכל היותר n.

הוכחה: אינדוקציה על n. ■

משפט 8.11 (משפט שוורץ־זיפל, Schwartz־Zippel) אם $p \not\equiv 0$ פולינום ב־ n משתנים מדרגה d מעל $\mathbb{Z}_p$, ו־ $r_1, \ldots, r_n \sim \mathcal{U}(\mathbb{Z}_p)$, אז

$$\Pr[p(r_1, \ldots, r_n) = 0] \leq \frac{d}{p}$$

הוכחה: אינדוקציה על n. עבור $n = 1$ זה נכון, כי ל־ p יש לכל היותר d שורשים, ולכן ההסתברות שבחרנו שורש באופן אחיד היא לכל היותר $\frac{d}{p}$.

נשים לב ש־ $p(x_1, \ldots, x_n) = \sum_i p'_i(x_2, \ldots, x_n) x_1^i$, כלומר אפשר להסתכל על p כפולינום במשתנה x_1 שהמרכיבים שלו הם פולינומים מעל $x_2, \ldots, x_n$. הפולינומים $\{p'_i\}$ יכולים להיות זהותית אפס או לא; ניקח את k להיות ה־ i המקסימלי כך ש־ $p'_i \not\equiv 0$. בפרט, $p'_k \not\equiv 0$ והדרגה שלו היא לכל היותר $d - k$ (כי ה־ x_1^k "לקח" k מהדרגה המקסימלית של p כולו).

לפי הנחת האינדוקציה, $\Pr[p'_k(r_2, \ldots, r_n) = 0] \leq \frac{d-k}{p}$. נתבונן בפולינום במשתנה אחד $q = p(x_1, r_2, \ldots, r_n)$ ונניח ש־ $p'_k(r_2, \ldots, r_n) \neq 0$. אז הפולינום q הוא מדרגה k. לפי מקרה הבסיס שראינו, מתקיים $\Pr[p(x_1, r_2, \ldots, r_n) = 0] \leq \frac{k}{p}$, כאשר ההסתברות היא על בחירת x_1.

כעת נכתוב נוסחת הסתברות שלמה עבור מה שאנחנו רוצים:

$$\begin{aligned}
\Pr[p(r_1, \ldots, r_n) = 0] &= \Pr[p(r_1, \ldots, r_n) = 0 \mid p'_k(r_2, \ldots, r_n) = 0] \cdot \Pr[p'_k(r_2, \ldots, r_n) = 0] + \\
&\quad \Pr[p(r_1, \ldots, r_n) = 0 \mid p'_k(r_2, \ldots, r_n) \neq 0] \cdot \Pr[p'_k(r_2, \ldots, r_n) \neq 0] \\
&\leq \Pr[p'_k(r_2, \ldots, r_n) = 0] + \Pr[p(r_1, \ldots, r_n) = 0 \mid p'_k(r_2, \ldots, r_n) = 0] \\
&= \frac{d-k}{p} + \frac{k}{p} = \frac{d}{p}
\end{aligned}$$

■

8.5 שקילות נוסחאות

הגדרה 8.12 אם F_1, F_2 נוסחאות אריתמטיות במשתנים $x_1, \ldots, x_n$, נאמר שהנוסחאות שקולות אם כל מקדמי המונומים בפולינומים המתאימים שווים.

הערה 8.13

1. לא נכון להסתכל על ערכי הצבות בנוסחאות, שכן אנחנו מדברים על פולינומים ולא על פונקציות. למשל, מעל $\mathbb{Z}_3$ הפולינומים 0 ו־ $x(x-1)(x-2)$ אינם שקולים כפולינומים, אבל כן שקולים כפונקציות.
2. דוגמא לשקילות: $(x_1 - x_2)(x_1 + x_2) \sim x_1 \cdot x_1 - x_2 \cdot x_2$.
3. כמובן, מתקיים ש־ $F_1 \sim F_2$ אמם $F_1 - F_2 \sim 0$.

הגדרה 8.14 בעיית שקילות הנוסחאות (equivalence) מוגדרת ע"י השפה

$$\mathrm{EQ} = \{F : \ F \sim 0\}$$

הערה 8.15 צריך לשים לב שפתיחת סוגריים והשוואת מקדמים תפעל בזמן אקספוננציאלי, כלומר $\mathrm{EQ} \in \mathrm{EXPTIME}$. בבירור, $\mathrm{EQ} \in coNP$ כי אפשר לבדוק הצבה בזמן פולינומיאלי (בתנאי שהמספרים שמציבים הם בגודל פולינומי באורך הנוסחא, או אם מסתכלים על זמן פולינומי באורך הנוסחא והמספרים שמציבים). אנחנו רוצים להשתכנע שאם $F \not\sim 0$ אז יש מספרים לא גדולים שאפשר להציב בנוסחא ולראות שהיא לא 0.

משפט 8.16 אם p פולינום ב־ n משתנים מעל שדה $\mathbb{F}$ ודרגתו $d \geq 0$, $p \not\equiv 0$, $S \subseteq \mathbb{F}$ קבוצה סופית, ונגריל $z_1, \ldots, z_1 \sim \mathcal{U}(S)$, אזי מתקיים

$$\Pr[p(z_1, \ldots, z_n) = 0] \leq \frac{d}{|S|}$$

הוכחה: זוהי וריאציה קלה ביותר על משפט זיפל־שוורץ שהוכחנו קודם. ■

מסקנה 8.17 קיימת הצבה כנ"ל במספרים נמוכים יחסית, למשל עבור $S = \{1, \ldots, n^5\}$ ולכן $\mathrm{EQ} \in coNP$ כפי שרצינו.

<u>אלגוריתם ל־ EQ</u>:
בהינתן נוסחא F באורך n מעל השדה $\mathbb{Q}$, נציב ב־ $\mathbb{F}$ ערכים רנדומיים מתוך הקבוצה $S = \{1, \ldots, n^5\}$ ונקבל אסם הערך של הנוסחא עבור ההצבה שבחרנו הוא 0.
<u>שלמות</u>: אם $F \sim 0$ אז $\Pr[\text{"yes"}] = 1$.
<u>נאותות</u>: אם $F \not\sim 0$ אז

$$\Pr[\text{"yes"}] \leq \frac{\deg(p_F)}{|S|} \leq \frac{n}{n^5} = \frac{1}{n^4} \leq \frac{2}{3}$$

<u>זמן הריצה</u>: הצבה אחת, ולכן פולינומיאלי.
מכאן אנו מקבלים ש־ $\mathrm{EQ} \in RP$.
כמובן, לא לכל פולינום יש נוסחא בגודל סביר שמייצגת אותו - משיקולי ספירה. לנוסחא יש הרבה פחות "דרגות חופש". כך שהתוצאה שקיבלנו מעניינת, אבל לא ישימה לכל פולינום.

הגדרה 8.18 יהיו F_1, F_2 נוסחאות אריתמטיות במשתנים $x_1, \ldots, x_n$. נאמר ש־ $F_1 \sim_B F_2$ (שקולה בוליאנית) אם $F_1(z_1, \ldots, z_n) = F_2(z_1, \ldots, z_n)$ עבור כל $z_1, \ldots, z_n \in \{0, 1\}$.

הערה 8.19

$$F_1 \sim_B F_2 \iff F_1 - F_2 \sim_B 0 \iff (F_1 - F_2)^2 \sim_B 0 \iff$$
$$\iff \sum_{u_1, \ldots, u_n \in \{0,1\}} (F_1 - F_2)^2(u_1, \ldots, u_n) = 0$$

הגדרה 8.20

$$\mathrm{BEQ} = \{F : \ F \text{ is a formula in } n \text{ variables}, \sum_{u_1, \ldots, u_n \in \{0,1\}} F^2(u_1, \ldots, u_n) = 0\}$$

הערה 8.21 אם היינו יודעים להכריע את BEQ היינו מקבלים בחינם גם מכונה להכרעת הבדיקה האם שתי נוסחאות שקולות בוליאנית. ברור ש־ $\mathrm{BEQ} \in \mathrm{EXPTIME}$ כי אפשר לעבור על 2^n הצבות בוליאניות ולבדוק בזמן פולינומי כל אחת - אבל אנחנו רוצים משהו יותר טוב. ברור גם ש־ $\mathrm{BEQ} \in coNP$ כי העד הוא פשוט הצבה בוליאנית, וקל לבדוק אותו.
לא ידוע היום האם $\mathrm{EQ} \in P$ והאם $\mathrm{BEQ} \in NP$. לא ידוע גם האם יש אלגוריתם רנדומי פולינומי. נשים לב שאותו רעיון כמו קודם לא עובד כי בחירת הצבה בוליאנית לא יכולה להשתמש במשפט שוורץ־זיפל.

<u>פרוטוקול אינטראקטיבי להכרעת</u> BEQ:[8]

הפרוטוקול הוא בין V המוודא ל־ P המוכיח, כאשר V שואל שאלות ו־ P עונה עליהן.

מספר השלבים בפרוטוקול יהיה ליניארי ב־ n (ספציפית, $n+2$), כאשר V מבצע חישוב פולינומי ב־ n בכל שלב, ולכן סך זמן הריצה שלו יהיה פולינומי.

P יכול לעשות חישוב כלשהו, והתשובות שלו הן באורך פולינומי.

<u>שלמות</u>: אם P, V עוקבים אחרי הפרוטוקול אז בסופו של דבר V מחזיר את המספר $s = \sum_{u_1,\ldots,u_n\in\{0,1\}} F(u_1,\ldots,u_n)$.

<u>נאותות</u>: אם V עוקב אחרי הפרוטוקול, אז לכל אסטרטגיה של P ההסתברות ש־ V מחזיר מספר שאינו s תהיה קטנה או שווה ל־ $\frac{1}{4}$.

כלומר, לא מובטח ש־ P "מתנהג יפה" אבל זה לא יקלקל יותר מדי - או ש־ V בכל מקרה ייתן את s, או שהוא יגיד "אני לא מוכן לדבר עם ה־ P הזה".

1. V שואל: "מהו s?", P עונה: "$s^{(?)}$"

נגדיר את הפולינום $p_1(x_1) = \sum_{u_2,\ldots,u_n\in\{0,1\}} F(x_1,u_2,\ldots,u_n)$, פולינום במשתנה אחד מדרגה $\geq n$. אנחנו לא יודעים לחשב אותו - רק מגדירים אותו.

2. V שואל: "מהו $p_1(x_1)$?", P עונה: "$p_1^{(?)}(x_1)$"

נשים לב ש־ p_1 הוא פולינום במשתנה אחד, ויש לו $n+1$ מקדמים לכל היותר. לכן אין בעיה לשלוח אותו מצד לצד - הבעיה היא רק לחשב אותו. נשים לב גם ש־ $s = p_1(0) + p_1(1)$.

כעת אם $n < \deg(p_1^{(?)}(x_1))$, אז V דוחה, וגם אם $p_1^{(?)}(0) + p_1^{(?)}(1) \neq s^{(?)}$, אז V דוחה.

לבסוף V בוחר $z_1 \sim \mathcal{U}(\{1,\ldots,n^5\})$. הרעיון הוא שאם $p_1^{(?)} \neq p_1$, אז בסיכוי גבוה גם $p_1^{(?)}(z_1) \neq p_1(z_1)$ בגלל משפט זיפל־שוורץ.

נגדיר את הפולינום $p_2(x_2) = \sum_{u_3,\ldots,u_n\in\{0,1\}} F(z_1,x_2,u_3,\ldots,u_n)$.

3. V שואל: "מהו $p_2(x_2)$?", P עונה: "$p_2^{(?)}(x_2)$"

אם $n < \deg(p_2^{(?)}(x_2))$ או $p_2^{(?)}(0) + p_2^{(?)}(1) \neq p_1^{(?)}(z_1)$, אז V דוחה.

כעת V בוחר $z_2 \sim \mathcal{U}(\{1,\ldots,n^5\})$.

נגדיר את הפולינום $p_3(x_3) = \sum_{u_4,\ldots,u_n\in\{0,1\}} F(z_1,z_2,x_3,u_4,\ldots,u_n)$.

4. V שואל: "מהו $p_3(x_3)$?", P עונה: "$p_3^{(?)}(x_3)$"

ושוב V מוודא את הדרגה ואת הסכום, וממשיכים כך כאשר האבחנה היא שאם p_k היה שקרי ו־ p_{k+1} היה אמיתי, הסיכוי להתלכדות הערכים בשלב הבא הוא קטן. כך המשכנו, וייתכן ש־ P שיקר לנו במשך כל הדרך. הגענו לשלב ה־ n:

n. בחרנו $z_{n-1} \sim \mathcal{U}(\{1,\ldots,n^5\})$ ומגדירים $p_n(x_n) = F(z_1,\ldots,z_{n-1},x_n)$.

V שואל: "מהו $p_n(x_n)$?", P עונה: "$p_n^{(?)}(x_n)$"

כעת V מוודא דרגה ומוודא ש־ $p_{n-1}^{(?)}(z_{n-1}) = p_n^{(?)}(1) + p_n^{(?)}(0)$.

n+1. בוחרים $z_n \sim \mathcal{U}(\{1,\ldots,n^5\})$ ואי אפשר להמשיך. עכשיו V יבדוק ש־ $p_n^{(?)}(z_n) = p_n(z_n) = F(z_1,\ldots,z_n)$ וידחה אם אין שוויון. אחרת, V יחזיר את $s^{(?)}$ שאנו חושבים שהוא s.

נשים לב שאין בעיה להעריך את $p_n(z_n)$ כי זהו פולינום במשתנה אחד - אין יותר סכום של מספר אקספוננציאלי של מונומים.

<u>שלמות</u>: קל לראות שהיא מתקיימת.

<u>נאותות</u>: רוצים להראות שאם $s \neq s^{(?)}$, אז V דוחה בדרך כלל, ואם הוא מקבל - זה קורה בהסתברות $\geq \frac{1}{4}$. ובכן, בשלב 1 V בודק ש־ $s^{(?)} = p_1^{(?)}(1) + p_1^{(?)}(0)$. כלומר, אם P רוצה לרמות הוא מוכרח להחזיר לנו $p_1^{(?)}$ שאינו p_1 האמיתי.

בשלב 2 היה לנו z_1 רנדומי, וכעת P נותן לנו את $p_2^{(?)}$ ושוב צריך להתקיים ש־ $p_2^{(?)}(0) + p_2^{(?)}(1) = p_1^{(?)}(z_1)$. כעת יש שתי אפשרויות:

- או ש־ $p_1(z_1) = p_1^{(?)}(z_1)$ - ההסתברות לזה היא $\geq \frac{1}{n^4}$, כי בחרנו את z_1 מתוך n^5 מספרים שונים והפולינום הוא מדרגה n;
- אחרת $p_1(z_1) \neq p_1^{(?)}(z_1)$, ואז P צריך לשקר ולהחזיר איזשהו $p_2^{(?)}$ שאינו p_2.

כך זה ממשיך, עד $p_n^{(?)}(x_n) \neq p_n(x_n)$ ובוחרים z_n רנדומי ובודקים האם $p_n^{(?)}(z_n) = p_n(z_n)$. אם עד עכשיו P שיקר, הסיכוי שהשוויון האחרון מתקיים הוא שוב $\geq \frac{1}{n^4}$.

כלומר, בכל שלב בפרוטוקול יש הזדמנות של $\geq \frac{1}{n^4}$ ש־ P יוכל לשקר ולשכנע את V. בסה"כ, ההסתברות של P לשקר בהצלחה לא תעלה על $\mathcal{O}(\frac{1}{n^3})$, וזה מה שרצינו מראש.

8.6 IP ו־ coNP

טענה 8.22 $\overline{\text{3SAT}} \leq_p \text{BEQ}$

הוכחה: יש אלגוריתם פולינומי אשר בהינתן נוסחא בוליאנית כלשהי φ מחזיר פולינום p_φ מעל אותם משתנים בדיוק כך שלכל השמה בוליאנית $(u_1,\ldots,u_n) \in \{0,1\}^n$ למשתנים $x_1,\ldots,x_n$ מתקיים $p_\varphi(u_1,\ldots,u_n) = \varphi(u_1,\ldots,u_n)$.

האלגוריתם פועל באינדוקציה. במקרה הבסיס $\varphi = 1$ נחזיר את הפולינום $p_\varphi = 1$.

עבור $\varphi = x_i$ נחזיר את הפולינום $p_\varphi = x_i$.

[8] שימו לב - לפי הודעת סגל הקורס, פרוטוקול זה אינו כלול בחומר למבחן.

עבור $\varphi = \overline{x_i}$ נחזיר את הפולינום $p_\varphi = (1 - x_i)$.
עבור $\varphi = \varphi_1 \wedge \varphi_2$ נחזיר את הפולינום $p_\varphi = p_{\varphi_1} \cdot p_{\varphi_2}$.
עבור $\varphi = \varphi_1 \vee \varphi_2$ נחזיר את הפולינום $p_\varphi = p_{\varphi_1} + p_{\varphi_2} - p_{\varphi_1} \cdot p_{\varphi_2}$.
מכאן ש־ φ אינה ספיקה אסם $\sum_{u_1,\ldots,u_n \in \{0,1\}} p_\varphi^2(u_1, \ldots, u_n) = 0$ וזו הרדוקציה. ■

מסקנה 8.23 $coNP \subseteq IP$

הוכחה: מיידית מהעובדה ש־ 3SAT היא NP־שלמה ולכן $\overline{\text{3SAT}}$ היא $coNP$־שלמה, וראינו ש־ $\text{BEQ} \in IP$. ■

8.7 שיתוף סוד

רוצים לחלק ל־ n אנשים חלקים של סוד, כך שרק כל k מתוכם יוכלו להרכיב את הסוד השלם, אבל ל־ $k-1$ מתוכם לא יהיה מה לעשות שהוא יותר טוב מאשר לנחש את הסוד.

אנו עובדים מעל שדה סופי $\mathbb{Z}_p$ עבור p ראשוני כלשהו. פורמלית, אם $D \in \mathbb{Z}_p$ הסוד שנבחר בהתפלגות אחידה, אנו רוצים ש־ k אנשים יוכלו לבצע חישוב פולינומיאלי שהתוצאה שלו היא D בוודאות, ואילו $k-1$ אנשים לא יוכלו בחישוב פולינומיאלי לקבל תוצאה שהיא יותר טובה מהתפלגות אחידה על $\mathbb{Z}_p$.

הדרישה ש־ D נבחר בהתפלגות אחידה לא מזיקה, כי במקרה שהיא לא, אפשר להגריל D' בהתפלגות אחידה ולהשתמש ב־ $D \oplus D'$ (xor, שהיא פעולה הפיכה) בתור הסוד. נדרוש גם $D, n < p$.

<u>הסכימה של שמיר</u>:

נגדיר פולינום $f(x)$ עם מקדמים $a_0 = D;\ a_1, \ldots, a_{k-1} \sim \mathcal{U}(\mathbb{Z}_p)$. זהו פולינום מדרגה $k-1 \geq$, ומתקיים $f(0) = D$. כעת נציב בפולינום $f(1), f(2), \ldots, f(n)$ וזה מה שמחלקים לאנשים - לאיש ה־ i נותנים את הזוג $(i, f(i))$.

כעת k מהאנשים יכולים לחלץ את D. בהינתן $(x_1, y_1), \ldots, (x_k, y_k)$ נגדיר את הפולינומים

$$l_i(x) = \prod_{m \neq i} \frac{x - x_m}{x_i - x_m}, \quad \deg(l_i) = k - 1$$

ומתקיים $l_i(x_i) = 1$ ולכל $j \neq i$ מתקיים $l_i(x_i) = 0$. את פולינום האינטרפולציה מקבלים כך:

$$\hat{f}(x) = \sum_{i=1}^{k} y_i l_i(x)$$

ואכן קל לוודא שלכל j, מתקיים $\hat{f}(x_j) = y_j$. כיוון שפולינום ממעלה $k-1 \geq$ נקבע ביחידות ע"י k נקודות, קיבלנו מתוך k הנקודות את f, כלומר $\hat{f} \equiv f$. כל שנותר לאנשים לעשות זה להציב $\hat{f}(0)$ ולקבל את D. כמובן, כל התהליך היה פולינומיאלי - מדובר בחיבור, חיסור, כפל וחילוק של מספרים ב־ $\mathbb{Z}_p$.

נראה גם של־ $k-1$ אנשים אין משהו טוב לעשות. כמה פולינומים שונים אפשר לקבל אם $x_1, \ldots, x_k$ קבועים ול־ $y_1, \ldots, y_k$ יש ערכים כלשהם? יש p^k זוגות שונים של $(x_1, y_1), \ldots, (x_k, y_k)$ כאלה. הפעלת אינטרפולציה על כל k־יה מתוך p^k ה־k־יות נותנת פולינום שונה מכל האחרים. למשל, בהינתן $(x_1, y_1), \ldots, (x_k, y_k)$ ו־ $(x_1, y_1'), \ldots, (x_k, y_k)$ מתקבלים $\hat{f}, \hat{f}'$ שונים, שהרי הם שונים אפילו על x_1. מכאן יש <u>לפחות</u> p^k פולינומים שונים מדרגה $k-1 \geq$ מעל $\mathbb{Z}_p$.

מצד שני, אפשר לספור את הפולינומים באמצעות המקדמים שלהם. יש k מקדמים שכל אחד מהם נבחר מ־ $\mathbb{Z}_p$ ולכן מספר הפולינומים השונים מדרגה $k-1 \geq$ הוא p^k <u>לכל היותר</u>. לכן, יש בדיוק p^k פולינומים שונים מעל $\mathbb{Z}_p$.

בהינתן רק $k-1$ זוגות, יש לנו את $y_1, \ldots, y_{k-1}$ בנקודות $x_1, \ldots, x_{k-1}$. נוסיף נקודה נוספת, $x_0 = 0$ ונבחר לו ערך כלשהו מ־ $\mathbb{Z}_p$, כלומר איזה $d = y_0 \sim \mathcal{U}(\mathbb{Z}_p)$. כל מספר שנבחר יספק פולינום אחר. כעת מכיוון שבחירת $a_0, \ldots, a_{k-1}$ הייתה בהתפלגות אחידה, הסיכוי ש־ $y_0 = d$ זהה לסיכוי שבחרנו פולינום ספציפי כלשהו, כלומר לא למדנו שום דבר על y_0, כנדרש.