סיכומים למבחן בקורס מודלים חישוביים

סמסטר א' 2008-9 (פרופ' נחום דרשוביץ)

חלק ראשון: חישוביות

בעיות חשיבות:

דוגמאות לפונקציות לא חשיבות:

פונקציה	תיאור	הערות
$bb(n)$ הבונה החרוץ	$max\{[p()] \mid p \in L_0, \lvert p\rvert \leq n\}$ בהינתן מספר n, מה הוא הפלט הגדול ביותר (מספרי) שתוכנית באורך לכל היותר n כלשהי יכולה ליצור.	
$mm(n)$	$min\{k \in \mathbb{N} \mid \forall p \in L_0, \lvert p\rvert \leq n \Rightarrow [p()] \neq k\}$ המספר הקטן ביותר שאינו חשיב ע"י כל תוכנית ללא קלט שאורכה קטן או שווה ל-n.	הוכחת אי חשיבות: אותו רעיון: מניחים בשלילה שקיימת תוכנית כזו מאורך N, וכותבים תוכנית $c()$ בה מוגדרת M ומחזירה את הפלט $M(5N)$ - פלט שלא אמורה להיות מסוגלת להחזיר.
$minProg(n)$	$min\{\lvert p\rvert \mid p \in L_0, [p()] \geq n\}$ אורך התוכנית ללא קלט הקטן ביותר, של תוכנית המחשבת מספר גדול או שווה ל-n.	הוכחת אי חשיבות: עושים רדוקציה ל-bb, ע"י הגדרת bb להחזיר את ה-k כך ש-$minProg(k) \leq n, minProg(k+1) > n$.
$Halt(< p, x >)$	מחזירה T אמ"מ p עוצרת על הקלט x.	הוכחת אי חשיבות (כריעות): עושים רדוקציה מ-$minProg$ ע"י ריצה על כל תוכנית ללא קלט אפשרית, בדיקה האם עוצרת, ואם כן – האם הפלט שלה גדול או שווה ל-n.

מושגים:

- חשיבות: פונקציה תהיה חשיבה אמ"מ ניתן לכתוב תוכנית מחשב (במודל חישובי כלשהו) המחשבת אותה. כיוון שמספר התוכניות הוא בן מניה, בעוד שמספר הפונקציות לא בן מניה, אזי לא כל הפונקציות חשיבות.
- כריעות: פונקציה תהיה כריעה אמ"מ היא חשיבה והטווח שלה הוא $\{T, F\}$, כלומר ניתן לחלק את כל הקלטים שלה לשתי קבוצות לפי הפלט שלהם (פרדיקט). פונקציות שהטווח שלהן הוא $\{T, F, \perp\}$ מגדירות שפות.

רדוקציה:

נאמר כי קיימת רדוקציה מ-f אל g, נסמן $f \leq g$ (או: $f \propto g$), אם באמצעות תוכנית המחשבת את g, ניתן לכתוב תוכנית המחשבת את f (יותר "קשה" מ-f; כלומר ניתן לכתוב את f כך: $(f := (some\ manipulation\ on\ g)$). אם $f \leq g$ אזי:

- g חשיבה $\Leftarrow f$ חשיבה; בבעיות הכרעה: g כריעה $\Leftarrow f$ כריעה.
- f אינה חשיבה $\Leftarrow g$ אינה חשיבה; בבעיות הכרעה: f לא כריעה $\Leftarrow g$ לא כריעה.

STS (State transition system):

מכונת מצבים מתארת אלגוריתם, כאשר מעבריה הן פונקציות חלקיות. כל מצב מחזיק את כל המידע הנדרש ע"מ להמשיך בחישוב.

STS היא רביעיה סדורה: $S = (Q, I, F, \delta)$, כך ש:

- Q: קבוצת מצבים סופית / אינסופית.
- $I \subset Q$: קבוצת מצבי התחלה.
- $F \subset Q$: קבוצת מצבי סיום.
- $\delta \subset Q \times Q$: יחס מעבר.

ריצה / חישוב: סדרה סופית / אינסופית של איברים מ-Q: $q_0 \to q_1 \to \cdots$ המקיימת: $q_0 \in I, \forall i: < q_i, q_{i+1} > \in \delta$.

ריצה מקבלת: ריצה סופית $q_0 \to \cdots \to q_n$ המקיימת $q_n \in F$.

בעיות הכרעה:

בעיות הכרעה: פרדיקטים, בעיות שיש להן תשובה T או F. $f:\Sigma^* \to \{T,F\}$, $L(f)=\{x \mid x\in\Sigma^*, f(x)=T\}$.

מושגים:

- א"ב: יסומן Σ; קבוצה **סופית** של תווים וסימנים.
- מחרוזת (מעל א"ב): סדרה **סופית** של תווים מ-Σ כלשהו. למשל: ε - המחרוזת הריקה.
- שפה: קבוצה סופית / אינסופית של מחרוזות. למשל: Σ^* - קבוצת כל המחרוזות מעל א"ב Σ.
- בעיה / שפה כריעה: L תהיה כריעה אמ"מ קיימת תוכנית חשיבה (עוצרת בכל מקרה) p כך ש: $p(x)=\begin{cases}T, x\in L\\F, x\notin L\end{cases}$. כל שפה **סופית** היא בפרט כריעה. כמו כן כל הבעיות עם מספר סופי של שאילתות, או: מספר סופי של קלטים שהתשובה אליהן היא כן, או: מספר סופי של קלטים שהתשובה אליהן היא לא, הן כריעות.
- בעיה / שפה כריעה למחצה: L תהיה כריעה למחצה אם קיימת תוכנית p (לא בהכרח עוצרת תמיד) כך ש: $p(x)=\begin{cases}T, x\in L\\F/\perp, x\notin L\end{cases}$. לכל הפרדיקטים p נסמן את $L(p)=\{x \mid p(x)=T\}$, L היא השפה ש-p מקבלת / מכריעה למחצה.
- מחלקה: קבוצה סופית / אינסופית של שפות.
- המחלקה R: מחלקת כל השפות הכריעות, ידועה גם בשם מחלקת השפות הרקורסיביות. סגורה תחת פעולת המשלים: $L\in R \Leftrightarrow \bar{L}\in R$.
- המחלקה RE: מחלקת כל השפות הכריעות למחצה.
- המחלקה $coRE$: מחלקת כל השפות שהמשלימה שלהן היא ב-RE.

רדוקצית מיפוי:

נאמר כי קיימת רדוקצית מיפוי מהשפה A לשפה B, ונסמן $A\leq_m B$ אם קיימת פונ' חשיבה f כך ש: $x\in A \Leftrightarrow f(x)\in B$ (B שפה "קשה" יותר מ-A).

אותה f גם תקיים: $x\in\bar{A}\Leftrightarrow f(x)\in\bar{B}$. אם $A\leq_m B$, אזי:

- $A\in RE \Leftarrow B\in RE$ (בפרט: $A\in R \Leftarrow B\in R$). מכאן: $A\in coRE \Leftarrow B\in coRE$.
- $B\notin RE \Leftarrow A\notin RE$ (בפרט: $B\notin R \Leftarrow A\notin R$). מכאן: $B\notin coRE \Leftarrow A\notin coRE$.
- A לא כריעה $\Leftarrow B$ לא כריעה.

דוגמאות לשפות לא כריעות:

- $Halt=\{<p,x> \mid p\ halts\ on\ x\}$
- $Halt_\varepsilon=\{<p> \mid p\in L_0,\ p()\ halts\}$
- $equiv=\{<f,g> \mid f\equiv g\ (\forall x: f(x)=g(x))\}$
- $pconst=\{<g> \mid \exists n\forall x: g(x)=\perp \vee g(x)=n\}$

משפט רייס:

תהי $L_\varphi=\{<p> \mid p-$ לא תכונה סמנטית $\varphi\}$ שפה המקיימת:

- שפה של תוכניות.
- שפה לא טרוויאלית: קיימות $p_1\in L_\varphi, p_2\notin L_\varphi$.
- φ היא תכונה סמנטית, כלומר אם $p_1\equiv p_2$ (מחזירות לכל קלט שהוא את אותו פלט, לא משנה איך), אזי $p_1\in L_\varphi \Leftrightarrow p_2\in L_\varphi$ (שתיהן שייכות / לא שייכות יחד ל-L_φ).

אזי $L_\varphi\notin R$.

דוגמאות לשפות לא כריעות לפי רייס:

- $PConst_i=\{<p> \mid p\ halts\ on \leq i\ inputs\}$
- $Fin=\{<p> \mid p\ halts\ on\ a\ finite\ number\ of\ inputs\}$

שקילות סמנטית:

- אם L שפה סמנטית (כל התוכניות בה בעלות תכונה סמנטית כלשהי), אזי אם $p_1,p_2\in L$ לא בהכרח מתקיים ש: $p_1\equiv p_2$. למשל: נניח השפה L היא שפת כל התוכניות המחזירות כפלט מספר זוגי, $p_1=\lambda x.4, p_2=\lambda x.2$.
- לעומת זאת, אם $p_1\equiv p_2$ אזי $p_1\in L \Leftrightarrow p_2\in L$ - כלומר הן שייכות/לא שייכות ל-L יחד.

***Stepper*:**

תוכנית חשובה בה משתמשים בהוכחות הפועלת באופן הבא: $stepper(p,x,n)$ מחזירה T אמ"מ ריצת $p(x)$ עוצרת (באופן תקין) לאחר לכל היותר n צעדים (אחרת מחזירה F).

<u>משפט הרקורסיה</u>:

לכל תוכנית $f: baby \to baby$ (שפה של תוכניות) קיים קלט c כך ש: $c = f(c)$.

הוכחה: $c \coloneqq k(k); k = \lambda w.f\big(w(w)\big) \Rightarrow c \equiv \Big(\lambda w.f\big(w(w)\big)\Big)(k) = f\big(k(k)\big) = f(c)$.

<u>כריעות למחצה ואנומרביליות</u>:

אנומרטור לשפה:

עבור שפה L נגדיר $l: \mathbb{N} \to L$ אנומרטור שהוא פונקציה חשיבה שלכל $k \in \mathbb{N}$ מחזירה $x \in L$, ופונקציה זו היא **על** L ($\forall x \in L \exists k \in \mathbb{N}: l(k) = x$).

ניתן לבנות אנומרטור לשפה באמצעות ***Zigzag*** (שבלול): תוכנית הרצה עם *stepper* על טבלת הקלטים האפשריים (מסודרים לקסיקוגרפית למשל) אל מול מספר הצעדים, ומחזירה בכל פעם את הקלט ה-i שעבורו ה-*stepper* החזיר T.

טענות:

- $L \in RE \Leftrightarrow$ קיים ל-L אנומרטור על.
- $L \in R \Leftrightarrow$ קיים ל-L אנומרטור מונוטוני (כלומר אם $x > y$ לקסיקוגרפית, אזי $l(x) > l(y)$).
- לכל $L \in RE$ (אינסופית כמובן) קיימת תת שפה $L' \subset L$ כך ש-$L \in R$.

תכונות חשובות:

- $R = RE \bigcap coRE$.
- המחלקות $RE, coRE$ סגורות תחת איחוד וחיתוך (הוכחה פשוטה).
- $\bar{L} \in coRE \Leftrightarrow L \in RE$.
- מכאן שמתקיים: $L, \bar{L} \in RE \ (or\ coRE) \Leftarrow L \in R$; $L \in RE \bigwedge L \in coRE \Leftarrow L \in R$.

<u>סיכום הוכחות כריעות/כריעות למחצה/קו-כריעות למחצה:</u>

<u>הוכחת $L \in R$</u>:

- למצוא תוכנית חשיבה המכריעה את L.
- להראות ש-L סופית.
- רדוקציה (מציאת L' שידוע שהיא כריעה למצוא רדוקציה או רדוקצית מיפוי: $L \leq L' \ / \ L \leq_m L'$).
- להראות $L \in RE$ וגם $L \in coRE$ ($\bar{L} \in RE$). או לחילופין על $coRE$.

<u>הוכחת $L \notin R$</u>:

- שיטת הליכסון.
- רדוקציה או רדוקצית מיפוי.
- משפט *Rice*.
- שימוש ברעיון של משפט *Rice*.

<u>הוכחת $L \in RE$</u>:

- להראות תוכנית המכריעה למחצה את L (לא חשוב מה מחזירה / האם מתבדרת עבור $x \notin L$).
- רדוקצית מיפוי **ל**בעיה כלשהי ב-RE.
- להוכיח שקיים אנומרטור ל-L.
- להראות ש-$\bar{L} \in coRE$.

<u>הוכחת $L \notin RE$ (שונה מאשר $L \in coRE$)</u>:

- שיטת הליכסון.
- רדוקציית מיפוי **מ**בעיה **שאינה** ב-RE.
- להראות ש-$\bar{L} \in RE \backslash R$.

חישוב וקונפיגורציות:

קונפיגורציה: המצב הנתון של מודל חישובי כלשהו. בהינתן מודל חשובי וקוני ניתן יהיה להמשיך את החישוב (למשל: תמונת הזיכרון וכו').

חישוב / היסטוריה חישובית חוקית:

סדרת קונפיגורציות $(c_0, c_1, \dots, c_n)$ המקיימת:

- c_0 קונפיגורצית התחלה.
- (c_i, c_{i+1}) הוא מעבר חוקי לכל i.
- c_n היא קוני סיום (מתקבלת תשובה והחישוב מסתיים).

Checker:

$checker(p, x, c)$: תוכנית הבודקת האם $c = (c_0, \dots, c_n)$ היא היסטוריה חישובית חוקית של ריצת התוכנית p על הקלט x.

מכונת מונים:

מכונת מונים עם k רגיסטרים היא זוג סדור $CM = (\Lambda, R)$ כך ש:

- $\underline{\Lambda}$: סדרה סופית של הוראות $l_1, \dots, l_m$ כך ש-$l_i \in \{inc\ j, dec\ j, if\ j = 0\ goto\ l_s, halt \mid l_s \in \Lambda, 1 \leq j \leq k\}$.
- $\underline{R}$: סדרת $r_1, \dots, r_k$ רגיסטרים לא חסומים.

קונפיגורציה של מכונת מונים: $\{a_1, \dots, a_k, j \mid \forall i: a_i \in \mathbb{N}, l_j \in \Lambda\}$ - תמונת המצב של המכונה.

תכונות:

- מכונת 3 מונים יכולה לחשב כל מה שמכונת n מונים יכולה.
- מכונת 2 מונים יכולה לסמלץ מכונת $n \geq 3$ מונים באופן הבא: נניח ... p_1, p_2, p_3 היא סדרת המספרים הראשוניים. לכל אחד מ-n הרגיסטרים נבחר p_i שייצגו. נסמן את הרגיסטרים של CM_n ב-r_i, אז במכונת CM_2:
  - רגיסטר אחד ייצג את כל הרגיסטרים של CM_n כך: $x = p_1^{r_1} \cdot p_2^{r_2} \cdot \dots \cdot p_n^{r_n}$.
  - רגיסטר שני יהווה רגיסטר טמפוררי לחישובים.

  הסימולציה: כל $r_i + +$ ב-CM_n יסומלץ ע"י $x = x * p_i$ (הכפלה). לבסוף יהיה ניתן לפרק לגורמים כדי להוציא את הערכים האמיתיים.

<u>דוגמא לשפה הקשורה למכונות מונים</u>:

$L = \{< CM, n > \mid one\ of\ r_i\ reaches\ the\ value\ 1000n\}$: שפה זו **כריעה**. רעיון ההוכחה: לכל היותר יהיו $(1000n)^k \cdot m$ קונפיגורצויות בהן כל הרגיסטרים קטנים ממש מ-$1000n$. מריצים את מכונת המונים $(1000n)^k \cdot m + 1$ צעדים - אם בדרך אחד הרגיסטרים הגיע ל-$1000n$, מחזירים T. אחרת, נכנסנו ל-$loop$ איפשהו בדרך ולעולם אף רגיסטר לא יגיע ל-$1000n$.

מכונת טיורינג (*Turing Machine*):

מכונת טיורינג היא שביעיה סדורה: $M = (Q, \Sigma, \Gamma, q_0, q_a, q_r, \delta)$ כך ש:

- Q: קבוצה סופית של מצבים.
- Σ: א"ב של השפה כך שהתו הריק לא נמצא בו. קלט למכונה יהיה מא"ב זה.
- Γ: א"ב סופי של המכונה (מה שהמכונה יכולה לכתוב בעצמה) הכולל בתוכו לפחות את Σ ואת התו הריק.
- $q_0 \in Q$: מצב התחלה.
- $q_a \in Q$: מצב קבלה, כשמגיעים למצב זה המכונה עוצרת, הפלט הוא על הסרט.
- $q_r \in Q$ $(q_a \neq)$: מצב דחיה.
- $\delta \in Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$: כאשר ראש המכונה על אות מסוימת עם מצב מ-Q, הוא כותב אות כלשהי, זז ימינה או שמאלה ועובר למצב כלשהו מ-Q.

חישוב: מתחילים עם קלט כלשהו על הסרט (מורכב מ-Σ) ומ-q_0 וממשיכים עד הגעה ל-q_a/q_r.

קונפיגורציות:

במכונת טיורינג קונפיגורציה תהיה מה שכתוב על הסרט + המצב בו נמצאים + מיקום הראש בסרט. מכאן:

עבור N משבצות ראשונות בסרט המכונה, מספר הקוני המקסימלי האפשרי הוא: $|Q| \times |\Gamma|^N \times N$.

גרסאות מקבילות של TM בעלות אותו כוח חישוב:

- תחילת הסרט מסומן.
- אפשרות לראש הסרט להישאר במקומו לאחר כתיבה ($\{L, R, S\}$).
- כותבת פעם אחת בלבד על הסרט.
- סרטים מרובים.
- סרט שאינו חסום לא ימינה ולא שמאלה (ניתן לנוע בחופשיות לשני הכיוונים עד אינסוף).
- גרסאות לא דטרמיניסטיות של מכונות טיורינג (NTM).

שקילות כוח חישובי של מודלים שונים:

המודלים החישוביים הבאים שקולים:

- שפות תכנות (מחשב).
- שפות מכונה (RAM).
- מכונת טיורינג.
- מכונות מונים (עם 2 מונים לפחות).

חלק שני: סיבוכיות זמן ומקום:

הגדרות:

סיבוכיות זמן: מספר הצעדים הנדרש לביצוע אלגוריתם כפונקציה של הקלט. בניגוד לחק הראשון, כאן **המודל החישובי משנה** לסיבוכיות (למשל מ״ט עם שני סרטים עדיפה על מ״ט עם סרט אחד).

טענה: כל בעיה הניתנת לפתרון ע״י מ״ט מרובת סרטים בזמן $O(t(n))$, ניתנת לפתרון ע״י מ״ט עם סרט אחד בזמן $O(t(n)^2)$.

סיבוכיות מקום: המקום המקסימלי בו משתמשים במהלך ריצת אלגוריתם כפונקציה של הקלט. במ״ט: חסום ע״י מספר הצעדים הנעשים ע״י המכונה (באופן כללי: ע״י קבוע כלשהו).

סיבוכיות זמן ריצה:

המחלקה $DTIME(T(n))$:

עבור פונ׳ $T: \mathbb{N} \to \mathbb{N}$, מחלקה זו תהיה אוסף כל הבעיות (בעיות הכרעה) שניתן להכריע בזמן $O(T(n))$, כאשר n הוא אורך הקלט (יש לציין מודל על איזה מודל חישובי מדובר).

המחלקה P ($coP =$):

$P = \bigcup_{c\geq 1} DTIME(n^c)$: מחלקת כל הבעיות הניתנות להכרעה בזמן (לכל היותר) **פולינומיאלי**. הגדרה זו **אינה** תלויה בפרטי המודל, ואלגוריתם פולינומיאלי נחשב יעיל.

המחלקה NP:

מחלקת הבעיות שאי אפשר / לא ברור אם ניתן לפתור אותן, אך בהינתן עדות פולינומיאלית בגודל הקלט, ניתן לוודא בזמן פולינומיאלי (בגודל הקלט) האם פתרון כלשהו נכון. כלומר: קיים פולינום $p: \mathbb{N} \to \mathbb{N}$ ומ״ט הרצה בזמן פול׳ כך ש: $\forall x: x \in L \Leftrightarrow \exists w \in \Sigma^{p(|x|)}: M(x,w) = T$.

לא ידוע אם: $P = NP$,$coNP = NP$,$EXP = NP$.

המחלקה EXP ($coEXP =$):

$EXP = \bigcup_{c\geq 1} DTIME\left(2^{n^c}\right)$: מחלקת כל הבעיות הניתנות להכרעה בזמן (לכל היותר) אקספוננציאלי. מתקיים: $P \subset NP \subset EXP$.

המחלקה $NP\text{-}Hard$:

$NP - Hard = \{L \mid \forall L' \in NP: L' \leq_p L\}$: כלומר כל השפות כך שלכל שפה ב-NP קיימת רדוקציה פול׳ אליה.

המחלקה $NPComplete$ (NPC):

$NPC = NP - Hard \cap NP$: כל השפות ה-$NP\text{-}Hard$ שהן גם NP בעצמן.

רדוקציה (מיפוי) פולינומיאלית:

נאמר כי $A \leq_p B$ אם קיימת פונ' חשיבה <u>ופולינומיאלית</u> f כך ש: $x \in A \Leftrightarrow f(x) \in B$. אם $A \leq_p B$ אזי מתקיים:

- $A \in P, NP \Leftarrow B \in P, NP$ (בהתאמה).
- $B \notin P, NP \Leftarrow A \notin P, NP$ (בהתאמה).
- $B \in NP - Hard \Leftarrow A \in NP - Hard, NPC$. כדי להראות ש-$B \in NPC$ יש להראות גם ש-$B \in NP$ (לא מספיק ש-A מ-NPC).

דוגמאות לבעיות וסיווגן:

* <u>הגדרת *CNF*</u>: נוסחאות לוגיות המורכבות מפסוקיות (שרשור משתנים / משלימי משתנים (ליטרלים) ב-$\vee$) המחוברות ביניהן ב-$\wedge$.

* <u>ספיקות</u>: בעיה תהיה <u>ספיקה</u> אם קיימת הצבת ערכי אמת (ליטרלים) במשתני הנוסחא שיתנו לה ערך T (למשל $X \wedge \bar{X}$ לא ספיקה).

בעיות ב-*NP* וב-*NPC*	
SAT (*NPC*)	כל הנוסחאות הספיקות מצורת *CNF*. לא ידוע אם בעיה זו ב-*P*, אך בהינתן עד שהוא הצבה מספקת לנוסחא כלשהי, ניתן לודא שהנוסחה ספיקה.
3SAT (*NPC*)	כל הנוסחאות הספיקות מצורת *3CNF* (כל פסוקית מכילה 3 ליטרלים בדיוק).
Finite domain (CSP)	בעיות המורכבות מ-($\neg$, $\wedge$, $\vee$, $=$) עם תחום סופי (למשל $\{0,1\}$).
HamPath (*NPC*) ($\overline{HamPath} \notin NP$)	כל הגרפים G שקיים בהם מסלול המילטוני (מסלול פשוט העובר בכל צמתי הגרף בדיוק פעם אחת).
IS (Independent Set) (*NPC*)	כל הזוגות $< G, k >$ של גרף G ומספר טבעי k, כך שב-G קיימת קבוצת צמתים בגודל (לפחות) k שאין ביניהם אף קשת. ניתן להשתמש בקבוצת הצמתים ב-*IS* בגודל k כעדות.
Clique (*NPC*) ($\overline{Clique} \notin NP$)	כל הזוגות $< G, k >$ של גרף G ומספר טבעי k, כך שב-G קיימת קבוצת צמתים בגודל (לפחות) k בה כולם מחוברים לכולם. ניתן להשתמש בקבוצת הצמתים ב-*Clique* בגודל k כעדות. רדוקציה פול' פשוטה מ-*IS*: את כל זוגות הצמתים שלא היו מחוברים בקשת מחברים, ואת כל הקשתות המקוריות מסירים.
VC (Vertex Cover) (*NPC*)	כל הזוגות $< G, k >$ של גרף G ומספר טבעי k, כך שב-G קיימת קבוצת צמתים S בגודל (לכל היותר) k כך שלכל קשת (u, v) מתקיים: או ש-$u \in S$ או ש-$v \in S$. ניתן להשתמש ברדוקציה פול' פשוטה מ-*IS*: לוקחים את קבוצת הקודקודים המשלימה ל-*IS* ב-G, ובמקום k לוקחים $|V| - k$.
SubsetSum (*NPC*) ($\overline{SubsetSum} \notin NP$)	כל קבוצות המספרים $\{x_1, \dots, x_k, t\}$ כך שקיימת קבוצה חלקית מתוך $\{x_1, \dots, x_k\}$ שסכום איבריה שווה ל-t. עדות מתאימה תהיה קבוצה חלקית המקיימת את התכונה הנחוצה.
בעיות ב-*P*	
Primality	האם מספר הוא ראשוני.
Eulerian Path	כל הגרפים G שקיים בהם מסלול אויילר.
2SAT	כל הנוסחאות הספיקות בהן כל פסוקית מכילה 2 ליטרלים בדיוק.
בעיות ב-$NP - Hard \cap \overline{NP - Hard}$	
$IS \oplus Clique$	כל הזוגות $< G, k >$ של גרף G ומספר טבעי k כך ש: או שקיימת ב-G *IS* מגודל לפחות k, או שקיים ב-G *clique* מגודל לפחות k, אך לא שניהם יחד.

<u>סיבוכיות מקום</u>:

מודל בדיקת סיבוכיות מקום:

מכונת טיורינג עם שלושה סרטים (מודל השקול ל-TM רגילה):

- סרט קלט (בגודל N): לקריאה בלבד, ראש נע לשני הצדדים.
- סרט עבודה (בגודל S): קריאה וכתיבה, ראש נע לשני הצדדים. חישוב מקום: המקום בו השתמשנו על סרט זה כפונקציה של הקלט.
- סרט פלט: לכתיבה בלבד, ראש נע ימינה בלבד.

גודל קונפיגורציה במודל זה: $|Q| \times \underset{\text{עבודה}}{S \times |\Gamma|^S} \times N \times \underset{\text{קלט}}{|\Sigma|^N}$.

המחלקה $SPACE(s(n))$:

עבור פונ' $s: \mathbb{N} \to \mathbb{N}$ מחלקה זו תהיה אוסף כל השפות המוכרעות במקום $O(s(n))$.

המחלקה *PSPACE*:

$PSPACE = \bigcup_{c\geq 1} SPACE(n^c)$: אוסף כל השפות הניתנות להכרעה תוך שימוש <u>במקום</u> פולינומיאלי לגודל הקלט (ללא חשיבות לסיבוכיות הזמן).

בעיות ב-*PSPACE*	
QBF	SAT עם שימוש גם ב-$\forall, \exists$ (למשל: $\forall X. \exists Y. (X \vee Y)$). שפה זו נמצאת ב-***PSPACE-Complete***.
$Halt\text{-}LBA$	$\{< M, x > \mid M\ halts\ on\ x, M\ is\ LBA\}$: כאשר LBA הוא מ"ט (מהמודל הנתון) בו סרט העבודה חסום ע"י גודל הקלט.
$QSAT$	$\{\varphi \mid \varphi\ is\ a\ true-quantified\ boolean\ formula\}$: כל הנוסחאות בהן אין משתנים חופשיים (כולם תחת כמתים בתחילת הנוסחה) שיש להן הצבה הנותנת ערך T. סיבוכיות מקום פולי' נובעת מפתרון בשיטה רקורסיבית.

<u>חלק שלישי: שפות רגולריות ואוטומטים, שפות חסרות הקשר</u>

<u>אוטומטים סופיים דטרמיניסטיים (*DFA*)</u>:

אוטומט שהזיכרון שלו סופי, כלומר בעל מספר סופי של מצבים. הגדרה פורמלית: $A = (Q, \Sigma, \delta, q_0, F)$ כך ש:

- <u>Q</u>: קבוצה סופית של מצבים.
- <u>Σ</u>: א"ב קלט סופי. ברירת המחדל תהיה: $\{0,1\}$.
- <u>$\delta: Q \times \Sigma \to Q$</u>: פונקצית מעבר.
- <u>$q_0 \in Q$</u>: מצב התחלתי.
- <u>$F \subset Q$</u>: קבוצת מצבי סיום (קבלה).

ההבדלים ממכונת טיורינג:

- אין ראש הנע שמאלה או ימינה על הקלט, אלא פשוט קוראים את הקלט תו אחרי תו ומתקדמים במצבים בהתאם.
- אין כתיבה.
- אין מצבי עצירה (עוצרים כאשר מגיעים לסוף הקלט: אם עצרנו במצב מ-F, נגיד שהאוטומט מקבל את אותו קלט).

שפה של אוטומט דטרמיניסטי: נאמר כי L היא השפה של A כאשר: $L(A) = \{w \in \Sigma^* \mid A\ accepts\ w\}$.

<u>אוטומטים סופיים לא דטרמיניסטיים (*NFA*)</u>:

אוטומט לא דטרמיניטי N יהיה: $N = (Q, \Sigma, \delta, q_0, F)$ כך ש:

- <u>$\delta: Q \times \Sigma \cup \{\varepsilon\} \to P(Q)$</u>: פונקצית המעבר יכולה לכלול בתחום שלה גם את התו הריק, כלומר לבצע מעבר "על ריק", והטווח שלה אינו יחיד אלא יכול להיות כל אחד מהמצבים השייכים לקבוצה חלקית כלשהי של מצבים מ-Q.
- <u>חישוב מקבל</u>: רצף תווים $w = w_1 w_2 \dots w_n$ ($w_i \in \Sigma \cup \{\varepsilon\}$) וסדרת מצבים $q'_0, \dots, q'_m$ כך ש:
  - $q_0 = q_0{}'$
  - $q_{j+1} \in \delta(q_j, w_{j+1})$
  - $q'_m \in F$

  כלומר ישנו <u>איזשהו</u> מסלול מקבל ב-N עבור המילה w.

שפה של אוטומט לא דטרמיניסטי: $L(N) = \{w \mid N\ accepts\ w\}$.

הפיכת *NFA* ל-*DFA*:

- <u>נפטרים ממעברי ε</u>: נניח יש לנו מעבר $q_j \xrightarrow{\varepsilon} q_k$, אזי נוריד מעבר זה ונשים מעברים בין q_j ישירות לכל אחד מהמצבים אליהם עובר q_k, בהתאמה. כמו כן, אם q_k מצב מקבל, נהפוך את q_j גם למקבל (אם היה מקבל קודם, יישאר מקבל עכשיו).
- <u>במקום מצב בודד נחזיק קבוצות מצבים</u>:
  - במקום q_0 יהיה $\{q_0\}$.
  - לכל מעבר ב-NFA ע"י $a \in \Sigma$ כלשהו, נניח ל-$q_1, \dots, q_k$, ניצור ב-DFA מעבר למצב: $\{q_1, \dots, q_k\}$, כאשר אם אחד מהמצבים בקבוצה זו הוא מקבל ב-NFA, המצב החדש הזה שהוגדר ב-DFA יהיה גם כן מקבל.

- ניתן לאחר מכן להוריד את כל המצבים שלא ניתן להגיע אליהם (למשל מצב שנכנסו אליו רק קשתות ε).

הפיכת מספר *DFA* ל-*NFA*:

ניתן פשוט להוסיף מצב חדש s שיהיה מצב ההתחלה ב-NFA עם מעברי ε לכל אחד מ-k מצבי ההתחלה המקוריים של כל אחד מה-DFA. כך יתקיים שקיים <u>ניחוש</u> (כנדרש) של מסלול בו $w \in L(A_1) \cup \dots \cup L(A_k)$ תתקבל ע"י ה-NFA שבנינו.

<u>מחלקת השפות הרגולריות</u>:

מחלקת השפות המתקבלות ע"י NFA או DFA זהות, והיא נקראת מחלקת השפות הרגולריות.

למת הניפוח לשפות רגולריות:

לכל שפה רגולרית L קיים קבוע ניפוח $p > 0$ כך שלכל מילה $w \in L, |w| \geq p$ ניתן לחלק את w ל-x, y, z כך ש:

- $|y| \geq 1$
- $|xy| \geq p$
- לכל $i \geq 0$: $xy^i z \in L$ (z יכולה להיות ε).

סגירות שפות רגולריות:

- חיתוך, איחוד, שרשור, משלים, $KleenStar$.
- $DropChar(L) = \{xy \mid x, y \in \Sigma^*, a \in \Sigma, xay \in L\}$ (הורדת תו אחד כלשהו מהמילים ב-L).
- $Rot(L) = \{xy \mid yx \in L\}$ (רוטציה של המילה; לכל מילה $n-1$ רוטציות אפשריות).

מינימליזציה של אוטומט:

- מוחקים מצבים לא נחוצים (למשל שלא ניתן להגיע אליהם).
- לכל מצב q נגדיר $L(q)$ כשפה המתקבלת החל ממצב זה (כאילו הוא q_0 ב-$L(q)$). את כל המצבים ששפות ההמשך שלהן שוות נאחד.

ביטוי רגלורי:

- פעולת האיחוד ($+$): $a + b$ משמעו שאותו חלק במילה יהיה אחד מהביטויים באיחוד.
- פעולת השרשור ($\cdot$): משמעו שרשור של שני ביטויים רגולרים.
- פעולת ה-$KleenStar$ ($*$): אותו ביטוי יכול לחזור $i \geq 0$ פעמים.

למשל: $(a + b)^* \cdot (c + d) = \{c, d, ac, ad, bc, bd, abc, abd, aababbc, \dots\}$. לכל אוטומט ניתן לבנות ביטוי רגולרי, ולהיפך.

<u>שיטות הוכחה</u>:

הוכחת רגולריות:

- הצגת ביטוי רגולרי לשפה.
- הצגת אוטומט (NFA או DFA) המקבל את השפה.
- הגעה לשפה דרך פעולות משמרות רגולריות משפות הידועות כרגולריות.
- שפות סופיות.

הוכחת אי-רגולריות:

- סתירה ללמת הניפוח: מראים מילה שאמורה להיות בשפה ומקיימת את תנאי למת הניפוח. מראים לשכל חלוקה שהיא קיים איזשהו i עבורו המילה המתקבלת לאחר ניפוח לפי אותו i (או כיווץ) אינה בשפה.
- הגעה משפה זו לשפה שידוע שאינה רגולרית ע"י פעולות משמרות רגולריות.

<u>בעיות כריעות לשפות רגולריות</u>: שייכות ($w \in L(A)$?), *emptiness* ($L(A) = \emptyset$?), *fullness* ($L(A) = \Sigma^*$?), *subset* ($L(A_1) \subset L(A_2)$?), *equivalence* ($L(A_1) = L(A_2)$?) ועוד.

דקדוקים חסרי הקשר:

דקדוק חסר הקשר הוא רביעייה סדורה: $G = (V, \Sigma, S, R)$ כך ש:

- V: קבוצה סופית של משתנים.
- Σ: קבוצה סופית של טרמינלים (תוים שאינם משתנים, שפה) ($V \cap \Sigma = \emptyset$).
- $S \in V$: משתנה התחלה.
- $R: V \to (V \cup \Sigma)^*$: אוסף חוקי גזירה.

גזירה:

מתחילים במשתנה S, ובכל שלב בוחרים איזשהו משתנה $A \in V$ וכלל גזירה $(A \to \beta) \in R$ ומחליפים את A ב-β. מסיימים כשיש רק טרמינלים.

טענה: $regular \subset CFL$ - השפות הרגולריות מוכלות במחלקת השפות ח"ה.

שפה של דקדוק חסר הקשר (*CFL*):

$L(G) = \{w \in \Sigma^* | \ S \Rightarrow^* w\}$: כלומר שפת כל המילים כך שקיים ב-G רצף גזירות (המתחיל ב-S) אשר בסופו ניתן להגיע למילה w.

למת הניפוח לשפות ח"ה:

תהי L שפה ח"ה, אזי יש לה קבוע ניפוח $p > 0$ כך שלכל $|w| \geq p, w \in L$ קיימת חלוקה ל-$w = uvxyz$ כך ש:

- $|vy| \geq 1$
- $|vxy| \leq p$
- לכל $i \geq 0$ מתקיים: $uv^ixy^iz \in L$

סגירות שפות ח"ה:

- איחוד, שרשור, $KleenStar$.
- חיתוך עם **שפה רגולרית**.
- הצבה ח"ה: להציב במקום טרמינל סימן התחלה של CFG אחר.
- $Reverse$.

לא סגורות תחת:

- חיתוך עם שפה ח"ה.
- משלים.
- $DropMid(L) = \{xy \mid x, y \in \Sigma^*, |x| = |y|, a \in \Sigma, xay \in L\}$.

בעיות כריעות לשפות ח"ה: שייכות ($w \in L(G)$?), ריקנות ($L(G) = \emptyset$?).

בעיות לא כריעות לשפות ח"ה: מלאות ($L(G) = \Sigma^*$?).

דקדוקים חסרי הקשר לינאריים:

לינארי: כל כלל גזירה הוא מהצורה: $A \to x$ או $A \to xBy$, ($A, B \in V, x, y \in \Sigma^*$).

לינארי ימני: כל כלל גזירה הוא מהצורה: $S \to \varepsilon$ או $A \to a$ או $A \to aB$ ($A, B \in V, a \in \Sigma$).

לינארי שמאלי: מוגדר כמו ימני רק במקום $A \to aB$ יהיה $A \to Ba$ (הולך ונבנה שמאלה במקום ימינה).

טענה: אוסף כל הדקדוקים הלינאריים הימניים הוא מחלקת השפות הרגולריות (מכאן $reg \subset CFL$).

Chomsky Normal Form (CNF):

דקדוק ח"ה בו כל כלל הוא מהצורה: $A \to BC$, $A \to a$, $S \to \varepsilon$ עבור $A, B, C \in V, a \in \Sigma$, כאשר A יכול להיות סימן ההתחלה, אך B,C לא.

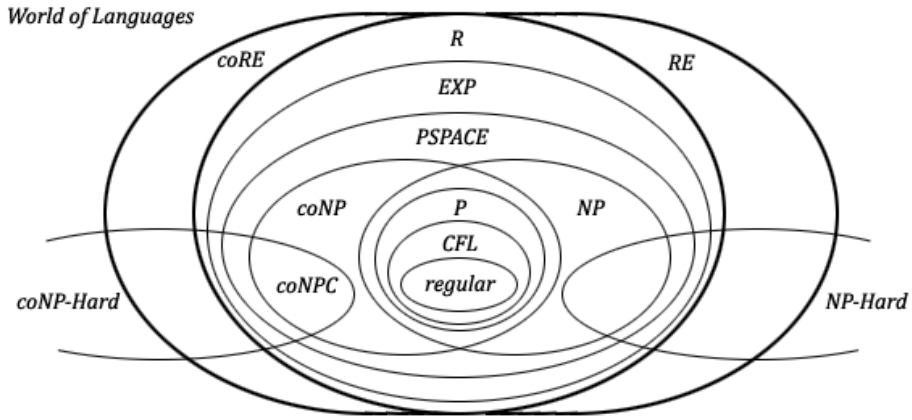
טענה: לכל שפה ח"ה L קיים דקדוק ח"ה G מצורת חומסקי.

שיטות הוכחה:

חוסר הקשר:

- מראים דקדוק ח״ה לשפה.
- השפה היא רגולרית או סופית.
- הגעה לשפה ע״י פעולות משמרות חוסר הקשר.

אי-חוסר הקשר:

- סתירה לחוסר הקשר באמצעות למת הניפוח.
- הגעה לשפה שידוע שאינה חסרת הקשר תוך שימוש בפעולות משמרות חוסר הקשר.

שרטוט כל מה שלמדנו: