<u>התמודדות עם בעיות "קשות": קירובים</u>

<u>דוגמה:</u>
מספר ההשמות המספקות את $f(\varphi) \triangleq \varphi$ עבור φ פסוק CNF.

<u>אבחנה 1:</u> אם $f \in POLY$ אז $SAT \in P$. (על מנת לבדוק האם $\varphi \in SAT$ פשוט נחשב את $f(\varphi)$ ואם הוא שונה מאפס אז נקבל ואחרת נדחה).

<u>הגדרה:</u> אלגוריתם A הוא אלגוריתם קירוב עבור פונקציה f עד כדי פקטור כפלי α, אם A הוא אלגוריתם יעיל, ומתקיים: $\frac{f(x)}{\alpha} \le A(x) \le f(x)\cdot\alpha$

<u>אבחנה 2:</u> אם ל f יש קירוב כפלי (יעיל), אז $SAT \in P$.

$$A(\varphi)=0 \Leftarrow f(\varphi)=0 \Leftarrow \varphi \notin SAT$$

$$A(\varphi)>0 \Leftarrow f(\varphi)>0 \Leftarrow \varphi \in SAT$$

לכן על מנת לבדוק בזמן פולינומי אם $\varphi \in SAT$ פשוט נחשב את $A(\varphi)$ ואם הוא שווה לאפס אז נדחה ואחרת נקבל.

<u>הגדרה:</u> אלגוריתם A הוא אלגוריתם קירוב עד כדי קבוע **חיבורי** d, ל f, אם לכל x מתקיים:

$$f(x)-d \le A(x) \le f(x)+d$$

<u>טענה:</u> אם ל f יש אלגוריתם קירוב חיבורי עד כדי קבוע d, והאלגוריתם יעיל, אז $SAT \in P$ (ולכן $P=NP$).

הוכחה: נסמן את אלגוריתם הקירוב ב A.
נראה אלגוריתם B להכרעה האם $\varphi \in SAT$.
אופן הפעולה של B על φ

1. מחשב את $k=\lceil \log_2(d)+2 \rceil$
2. $\varphi' = \varphi \wedge (y_1 \vee \overline{y}_1) \wedge (y_2 \vee \overline{y}_2) \wedge \ldots (y_k \vee \overline{y}_k)$ (עבור y-ים - משתנים שלא הופיעו ב φ)
3. חשב את $A(\varphi')$.
4. אם $2d \ge A(\varphi')$ אז דחה ואחרת קבל.

<u>אבחנות:</u>

1. B הוא אלגוריתם יעיל - חישוב הלוג לוקח פשוט מעבר על d ולכן קבוע. חישוב הנוסחה החדשה דורש זמן ליניארי - כתיבת הנוסחה הישנה + קבוע. הרצת A דורשת זמן פולינומי ע"פ הגדרתו.
2. $f(\varphi')=f(\varphi)\cdot 2^k$ - פשוט בוחרים y-ים כרצוננו. את המשתנים שב-φ מספקים באותו אופן כמו שמספקים אותם ב φ במקור.

אם $\varphi \notin SAT$ אז לא ניתן לספק את הפסוק, ולכן $f(\varphi)=0$. לכן, ע"פ אבחנה 2, $f(\varphi')=0$. לכן $A(\varphi') \le d$. לכן B תדחה. אם $\varphi \in SAT$ אז $f(\varphi)>0$. לכן, ע"פ אבחנה 2, $f(\varphi') \ge 2^k \ge 4d$ לכן $A(\varphi') \ge 3d$. לכן B תקבל.

המחלקה $CO-NP$: $CO-NP=\{L \mid \bar{L} \in NP\}$

<u>דוגמה:</u> $\overline{SAT}$ - אוסף כל הפסוקים מ CNF שלא ניתן לספק אותם.

$P \subseteq CO-NP \subseteq R$ (כי P סגורה לפעולת משלים)

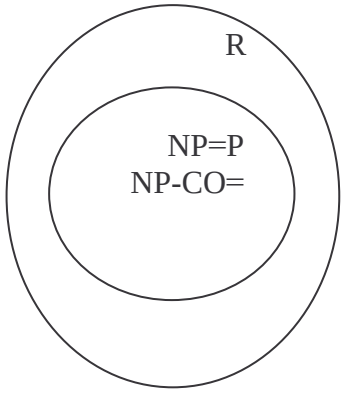
האם $P=CO-NP$? זה שקול לשאלה האם NP=P.

האם $NP=CO-NP$?
אם NP=P אז $NP=CO-NP$.
אם $P \neq NP$ זה לא גורר שום דבר לגבי השאלה האם $NP=CO-NP$.

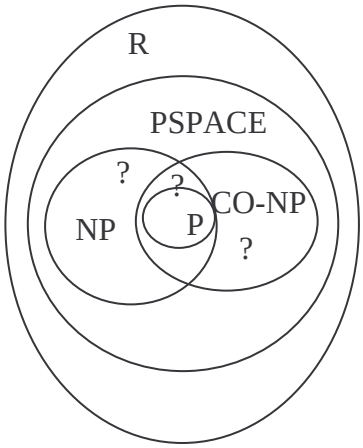
שפה $CO-NP$ שלמה L היא שפה המקיימת:
1. $L \in CO-NP$.
2. לכל $L' \in CO-NP$ מתקיים $L' \leq_P L$.

<u>אבחנה:</u> L היא שפה $CO-NP$ שלמה אם ורק אם $\bar{L}$ היא שפה NP שלמה.

אם NP=P



אם $NP \neq P$



<u>המחלקה PSPACE - סיבוכיות זיכרון:</u>
$S_M(x) \triangleq$ התא הימני ביותר בו M מבקרת בריצתה על M.
$PSPACE=\{L \mid$ L {יש מ"ט M בעלת סיבוכיות זיכרון פולינומית המקבלת את

<u>ידוע:</u>
1. $P \subseteq PSPACE$ - אם זמן הריצה הוא פולינומי, אז לא ניתן להתקדם יותר ממספר פולינומי של צעדים על הסרט.

2. $PSPACE \subseteq R$ - אם כמות הזיכרון חסומה, אז יש כמות חסומה של קונפיגורציות שניתן להיות בהן, ולכל מספיק לספור בכמה קונפיגורציות ביקרנו עד עכשיו, ואם עברנו ביותר מדי קונפיגורציות, אז אפשר לדעת שנכנסו ללולאה ולכן לדחות.

3. $NP \subseteq PSPACE$

הוכחות אפשריות:
א. המכונה שבנינו בהוכחת הטענה $NP \subseteq R$ מקיימת את הדרוש.
ב. ההגדרה השניה של NP : קיים יחס... $L = \{x \mid \exists y : (x, y) \in R\}$ - המכונה שמחפשת y בחיפוש מלא, מחזיקה זיכרון פולינומי.
ג. $SAT \in PSPACE$ - נעבור אחת - אחת על כל ההשמות, ונבדוק האם היא מספקת את הפסוק. צריך לשמור בזיכרון רק את ההשמה הנוכחית ואת הפסוק. לכן סה"כ הזיכרון פולינומי.
תהי $L \in NP$. נראה ש $L \in PSPACE$: תהי f_L הרדוקציה $L \leq_P SAT$.
אלגוריתם עבור L על קלט x:
- חשב את $\varphi = f_L(x)$ (זמן פולינומי ולכן זיכרון פולינומי).
- בדוק האם $\varphi \in SAT$ ע"י האלגוריתם שראינו.

<u>לא ידוע:</u>
האם $P = PSPACE$? האם כל דבר שניתן לפתור בזיכרון פולינומי, ניתן לפתור גם בזמן פולינומי.
האם $NP = PSPACE$?

טענה: קיימת שפה ב R שאיננה ב PSPACE.

תזכורת: "ראינו" $L \in R \setminus NP$
$NP \subseteq DTIME(2^{n^c})$ - כל שפה ב NP - יש לה מ"ט הרצה זמן אספוננציאלי.
קיימת $L \in R \setminus DTIME(2^{n^c})$
$PSPACE \subseteq DTIME(2^{n^c})$ (נובע מהוכחת $PSPACE \subseteq R$)
לכן $L \in R \setminus PSPACE$.

<u>הגדרה:</u> שפה L נקראת $PSPACE$-שלמה אם
א. $L \in PSPACE$
ב. לכל שפה $L' \in PSPACE$ מתקיים $L' \leq_P L$.

<u>מסקנה:</u> אם L היא שפה PSPACE שלמה, אז:
א. $P = PSPACE \Leftrightarrow L \in P$
ב. $NP = PSPACE \Leftrightarrow L \in NP$

<u>דוגמה 1:</u>
$BS = \{\langle M \rangle, x, 1^S \mid$ המכונה הדטרמיניסטית M מקבלת את x תוך שימוש בזיכרון קטן או שווה ל S$\}$
<u>טענה:</u> BS היא שפה PSPACE שלמה.
הוכחה:
א. $BS \in PSPACE$ - נסמלץ את ריצת M על x. אם M חורגת מזיכרון S עצור ודחה. אחרת עשה כמוה.
ב. לכל שפה $L \in PSPACE$ קיימת מ"ט דטרמיניסטית M, המקבלת את L תוך שימוש בזיכרון פולינומי $p(n)$.
הרדוקציה: $L \leq_p BS$: $f_L(x) = \langle M \rangle, x, 1^{p(|x|)}$

<u>דוגמה 2:</u>

SAT : נתון פסוק CNF, φ. האם φ הוא ספיק?

$$\psi = \exists x_1 \exists x_2 ... \exists x_m \varphi\left(x_1, x_2 ... x_m\right)$$

פסוק φ שייך ל SAT אם ערך האמת של ψ הוא TRUE.

הגדרה: נוסחאות Quantified Boolean Formula - QBF

נוסחת QBF היא נוסחה מהצורה:

$\psi = Q_1 x_1 Q_2 x_2, ..., Q_n x_n \varphi\left(x_1, x_2 .., x_n\right)$ כאשר φ היא נוסחת CNF ולכל i מתקיים $Q_i \in \{\exists, \forall\}$

השפה: $TQBF = \{\psi \mid$ ψ היא נוסחת QBF עם ערך אמת TRUE $\}$

משפט: TQBF היא PSPACE שלמה

הוכחה:

א. $TQBF \in PSPACE$

אלגוריתם: $EVAL$ על קלט ψ:

אם ψ נוסחה ללא כמתים אז חשב.

אחרת: $\psi = Q_j x_j Q_{j+1} x_{j+1}, ..., Q_n x_n \varphi\left(x_j, x_{j+1} .., x_n\right)$

נגדיר: ψ_T - הנוסחה המתקבלת מ ψ ע"י השמטת $Q_j x_j$ והצבת $x_j = TRUE$.

נגדיר: ψ_F - הנוסחה המתקבלת מ ψ ע"י השמטת $Q_j x_j$ והצבת $x_j = FALSE$.

רקורסיבית חשב: $a = EVAL\left(\psi_T\right)$, $b = EVAL\left(\psi_F\right)$

אם $Q_j = \forall$ החזר את $a \wedge b$.

אם $Q_j = \exists$ החזר את $a \vee b$.

נכונות: מהגדרת ערך האמת של נוסחת QBF.

זיכרון: עומק הרקורסיה קטן או שווה ל n ובכל רמה ברקורסיה, הזיכרון הדרוש הוא $O\left(|\psi|\right)$.

לכן סה"כ זיכרון: $O\left(|\psi|^2\right)$

ב. לכל $L \in PSPACE$ צ"ל: $L \leq_P TQBF$.

רעיון: דומה למשפט קוק.

בהשוואה למשפט קוק:

- המכונה כאן היא דטרמיניסטית.
- כאן הזמן אינסופי ולכן בפרט לא נוכל אפילו להחזיק משתנה לכל תא בטבלה.
- לרשותנו הכמת $\forall$