חישוביות

מחברת סיכום מתוך ההרצאות של דורון מור
אוניברסיטת אריאל 2022

חישוביות - מחברת הרצאות

נערך ונכתב על-ידי דור עזריה

2022

ספר זה לא נבדק על ידי מרצה, יתכן שימצאו טעויות.

לסיכומים נוספים שלי במדעי המחשב ומתמטיקה:
https://dorazaria.github.io/

מוזמנים לעקוב אחרי 😊:

LinkedIn: https://www.linkedin.com/in/dor-azaria/

GitHub: https://github.com/DorAzaria

חלק א' - חישוביות

חלק ב' - סיבוכיות

מבוא לקורס

חישוביות

בעיות שניתן לפתור על ידי מחשב. תורת החישוביות היא הבסיס למדעי המחשב, והיא עוסקת במודלים לחישוב ובפונקציות הניתנות לחישוב במסגרתם. השאלה הבסיסית בתורת החישוביות היא: **מה מחשבים יכולים לחשב, ומה לא?** בניגוד להנחה נפוצה, ישנן פונקציות שאי אפשר לחשב. בעיית העצירה מהווה דוגמה לפונקציה שכזו: ניתן להוכיח כי אין תוכנית היכולה לקבל כקלט תוכנית כלשהי וקלט לאותה התוכנית, ולהכריע האם התוכנית תעצור.

סיבוכיות

נחלק את עולם הבעיות שלנו לכמה מחלקות לפי סיבוכיות זמן הריצה שלהן. (או יעילות).
במדעי המחשב, סיבוכיות (באנגלית: *Complexity*) היא כלי מדד מתמטי של משאבי המערכת הנחוצים לפתרון בעיה נתונה באמצעות מחשב. המשאב העיקרי הנבחן הוא **זמן הריצה**, כלומר נבחן משך הזמן הנחוץ לשם ביצוע האלגוריתם. משאב נוסף הוא **הזיכרון** הנחוץ לשם ביצוע האלגוריתם. ניתן להביא בחשבון משאבים נוספים, כגון כמה **מעבדים** נחוצים לשם פתרון הבעיה בעיבוד מקבילי. התורה החוקרת סיבוכיות קרויה תורת הסיבוכיות. ענף הסיבוכיות נבדל מענף החישוביות, שבו נבחנת השאלה האם ניתן בכלל לפתור בעיה נתונה, בלא קשר לכמות המשאבים הנחוצה.

מודל חישובי

מודל חישובי הוא **מכונה תיאורטית** עם רכיבים מסוימים (למשל: גודל זיכרון, א"ב, סרט קלט). כשנדבר על מודלים חישוביים, נדבר על אלגוריתם כללי לפתרון הבעיה. באוטומטים ראינו מודלים חישוביים כגון אס"ד, אסל"ד ואוטומט מחסנית. בקורס זה המודל החישובי העיקרי יהיה **מכונת טיורינג**.

פסוקים

נעסוק בפסוק שהוא פונקציה בוליאנית בצורת **CNF** כמו למשל: $\left(x_1 \vee \bar{x_2} \vee x_4\right) \wedge \left(x_2 \vee \bar{x_3} \vee x_4\right)$.
"השמה מספקת" לפסוק מגדירה האם יש דרך להציב ערכי T/F בכל המשתנים האלה כדי שנקבל TRUE.

פסוק CNF

פסוק הוא **CNF** אם הפסוק כולו הוא "וגם" של הרבה פסוקים אחרים (פסוקיות).
כל פסוקית היא "או" של ליטרלים (משתנים אטומיים כמו למשל T או F).
נשים לב - השמה מספקת לפסוק CNF ⇔ היא מספקת כל פסוקית בו.

סוגי בעיות

ישנם שני סוגים של בעיות:

1. **בעיית קיימות** - אנחנו רוצים לדעת אם **קיים** פתרון.
2. **בעיית חיפוש** - אנחנו רוצים לדעת **מה** הפתרון.

אנו נעסוק בעיקר בבעיות הקיימות.

פתירות של בעיה

פתירות של בעיה היא ביחס למודל חישובי כלשהו, למשל מחשב או בן אדם (דף ועט).

בעיות לא פתירות - הכוונה שאין פתרון כללי לבעיה. כלומר, אם האלגוריתם מצליח לפתור מקרה פרטי בבעיה זה לא אומר שקיים פתרון כללי.

בעיות פתירות ניתן לסווג ל-"קשות" או "קלות". הסיווג נקבע לפי זמן הריצה *המינימלי* לפתרון הבעיה.
זמן הריצה הוא פונקציה של אורך הקלט. למשל עבור פסוק עם n משתנים, זמן הריצה הוא $O(n)$.

	בעיה קלה	בעיה קשה
הגדרה	בעיה היא קלה אם קיים אלגוריתם שיודע לפתור את הבעיה בזמן ריצה פולינומי.	בעיה היא קשה אם כל אלגוריתם צריך זמן ריצה לפחות אקספוננציאלי בגודל הקלט.
שיטתיות	מספיק להראות קיימות של אלגוריתם בעל זמן ריצה פולינומי ולהוכיח את זמן הריצה.	יש להוכיח שלא קיים אף אלגוריתם פולינומי שיפתור אותה.
דוגמה כללית ולא מספקת	בעיית המסלול הטוב ביותר - נפתור בעזרת אלגוריתם Floyd-Warshall ב-$O(n^3)$ שאכן בזמן ריצה פולינומי.	אלגוריתם שפותר בעיה מסויימת בזמן ריצה לפחות $O(2^n)$ (אקספוננציאלי). ולא קיים אלגוריתם יותר יעיל מזה.

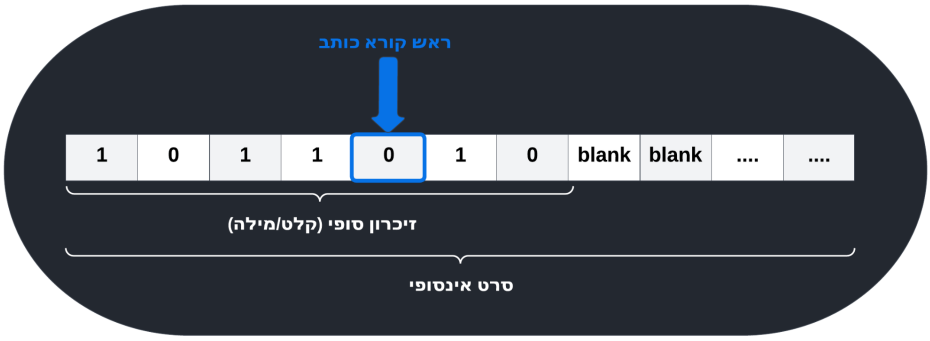
ייצוג בעיה על ידי שפה

מכונות טיורינג תקבל כקלט קידוד $< X >$ של האובייקט X עליו היא עובדת.
אובייקט X יכול להיות גרף, פסוק, פולינום ועוד.. והקידוד של כל אובייקט מסומן ב-$< X >$.
לדוגמה - עבור בעיית קשירות של גרף X, מכונת הטיורינג M תקבל כקלט את קידוד הגרף $< X >$ ואז תפעל על הקידוד הזה.

מכונת טיורינג

מבוא

מכונת טיורינג (Turing Machine) הוא מודל שהציע אלן טיורינג (אבי מדעי המחשב) טרם המצאת המחשב. המכונה מתוארת באופן דמיוני כסרט אינסופי של תאים וראש קריאה בעל זיכרון סופי היכול לקרוא את תוכנו של התא שמעליו הוא ממוקם, לכתוב באותו התא וכן לנוע ימינה או שמאלה על הסרט. המכונה מקבלת את הקלט שלה באמצעות הסרט, עליו היא יכולה גם לכתוב, ולזוז כרצונה לכל נקודה על גביו.



מכונת טיורינג (מ"ט) מתארת בצורה פורמלית-מתמטית, כיצד ניתן לבצע פעולות חישוביות שונות כגון זיהוי מילים השייכות לשפה פורמלית, ביצוע פעולות חיפוש ומיון בקלט ועוד, והיא למעשה האוטומט החזק ביותר לביצוע חישובים, ולמעשה לעיתים המושג "חישוב" מוגדר על סמך פעולות הניתנות לביצוע באמצעות מ"ט. התזה של צ'רץ'-טיורינג קובעת שכל חישוב או אלגוריתם בר-ביצוע, ניתן לתרגם למכונת טיורינג. מבחינה זו, מכונת טיורינג שקולה לכל מחשב.

רכיבי מכונת טיורינג

מכונת טיורינג M היא שביעייה $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$ כאשר:

- Q היא קבוצה סופית של מצבים כלשהם.
- q_0 מצב התחלתי.
- $F \subseteq Q$ הינה קבוצת המצבים הסופיים. המכונה רצה עד שמגיעה למצב סופי $q \in F$, במצב זה המכונה תחליט אם המילה התקבלה או לא התקבלה.
- $\bar{b}$ נקרא "blank" המסמן תא ריק בסרט (לפי הגדרה $\bar{b} \notin \Sigma$). אפשר לכתוב $\bar{b}$ גם באמצע הקלט.
- $\Sigma \subseteq \Gamma$ הינה א"ב הקלט. (בדר"כ $\Sigma = \{0, 1\}$).
- Γ הינה א"ב עבודה (בדר"כ $\Gamma = \{0,\ 1,\ \bar{b}\}$).
- δ היא פונקציית המעברים המוגדרת כך: $\delta\colon\ (Q\backslash F) \times \Gamma \to Q \times \Gamma \times \{L, S, R\}$.
  כלומר, הפונקציה קוראת: (מצב לא סופי ,אות מהסרט) ועוברת ל: (מצב, אות מהסרט, צעד).
- הקבוצה $\{L, S, R\}$ מכילה פעולות הזזה של הראש: $L = Left,\ S = Stay,\ R = Right$.

תכונות מכונת טיורינג

1. מ"ט M תעצור כשתגיע למצב סופי - אחרת, עלולה לרוץ עד אינסוף.
2. **פלט המכונה**: הוא מילה $y \in \Gamma^*$ שהיא רצף כל התווים על הסרט שמשמאל לראש.
3. בקורס הזה, האינדקס של התא הראשון של הסרט הוא "1" ולא "0" כמו במערכים מוכרים בתכנות.
4. אם מ"ט M לא עצרה על x, אזי הפלט שלה לא מוגדר. זה מאפשר חישוב של פונקציות "לא מלאות", כלומר פונקציות שלאוו דווקא מוגדרות לכל קלט.
5. ישנן מכונות טיורינג המחשבות:
   - <u>פונקציות</u> - נתמקד בפלט של המכונה.
   - <u>שפות</u> - נתמקד במצב $q \in F$ בו סיימנו (כאשר q הוא מצב מקבל או מצב דוחה/מגעיל). כלומר, לא מעניין אותנו מה מכיל הפלט - העיקר שהגענו למצב סופי.
6. **נסמן** את הפונקציה של מ"ט M בתור $f_M(x)$.

דוגמת בנייה למכונה המזהה שפה

בנו מכונת טיורינג $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$ המזהה את השפה: $x1 \in \{0,1\}^*$.

<u>פתרון משולב עם הסבר</u>: אנחנו רוצים לוודא שהתו האחרון הוא "1" ואז המילה מתקבלת.

אנחנו נקרא את המילה לאורך הסרט עד שנגיע ל-$\bar{b}$ הראשון ואז נחזור תא אחד אחורה ונקבע ש:

- אם כתוב בתא הזה "1" אז נעבור למצב מקבל $q_{accept} \in F$.
- אם כתוב בתא הזה "0" אז נעבור למצב דוחה $q_{reject} \in F$.

אנחנו נכתוב טבלת מעברים עבור המכונה M שמזהה את השפה $x1$:

- השורה הירוקה הם תווים $\gamma \in \Gamma$ שנמצאים בתוך התאים של הסרט שאותו אנחנו קוראים.
- העמודה הצהובה הם מצבים לא סופיים $Q \backslash F$ כלשהם שנטייל בהם במהלך הריצה.

לפי ההגדרה של δ, היא מקבלת **מצב נוכחי ואות קריאה** ומחזירה לנו: (צעד הזזה לראש, אות כתיבה, מצב).

$Q \backslash F$ \ Γ	**0**	**1**	$\bar{b}$
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_{reject}, 0, S)$
q_1	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_{accept}, 1, S)$

המצב q_0 אומר שהתא האחרון מכיל 0 ו-q_1 אומר שהתא האחרון מכיל 1. לכן, כשנקרא את ה-$\bar{b}$ הראשון ממצב q_1 אז נשלח למצב מקבל q_{accept} ואם נקרא את ה-$\bar{b}$ הראשון ממצב q_0 אז נשלח למצב דוחה q_{reject}.

פונקציה של מכונת טיורינג

לכל מ"ט M הפונקציה שלה מוגדרת כך: $f_M: \Sigma^* \to \Gamma^*$.

לכל מילה $w \in \Sigma^*$ הפונקציה $f_M(w)$ הינו הפלט של המכונה M בריצתה על הקלט w.

מכיוון שמ"ט לא תמיד עוצרת, הפונקציה f_M לא תמיד מלאה.

תזכורת - פונקציה מלאה היא פונקציה שמוגדרת לכל קלט.

דוגמה 1 - בנייה למכונת טיורינג המחשבת פונקצייה

המכונה M מקבלת x ורושמת $0x$ כלומר $f_M(x) = 0x$.

<u>רעיון הבנייה</u>: נרשום 0 בהתחלה, ובכל צעד נזכור האם לכתוב 0 או 1. בשונה מהדוגמה הקודמת עם השפה, כאן אין לנו עניין של מקבל או דוחה אלא מעניין אותנו הפלט של הפונקציה. אנחנו צריכים לסיים כשאנחנו מגיעים ל-$\bar{b}$ הראשון, ובדרך צריכים לדחוף כל תא צעד ימינה (shift right) ולכתוב "0" בתא הראשון.
לכן, התפקיד של המצבים q_0, q_1 ישמשו אותנו כדי לדעת מה לכתוב בתא הבא בסרט בכל צעד (הזזה).

<u>פורמלית</u>: $M = \left(Q = \{q_0, q_1, q_{end}\}, q_0, F = \{q_{end}\}, \Gamma = \{0, 1, \bar{b}\}, \Sigma = \{0, 1\}, \bar{b}, \delta\right)$.

אנחנו נכתוב טבלת מעברים עבור המכונה M שמזהה את הפונקציה $f_M(x) = 0x$.

- השורה הירוקה הם תווים $\gamma \in \Gamma$ שבתוך התאים של הסרט שאותו אנחנו קוראים.
- העמודה הצהובה הם מצבים לא סופיים $Q \backslash F$ כלשהם שנטייל בהם במהלך הריצה.

לפי ההגדרה של δ, היא מקבלת **מצב נוכחי ואות לקריאה** ומחזירה לנו: (צעד הזזה לראש, אות כתיבה, מצב).

$Q \backslash F$ \ Γ	**0**	**1**	$\bar{b}$
q_0	$(q_0, 0, R)$	$(q_1, 0, R)$	$(q_{end}, 0, R)$
q_1	$(q_0, 1, R)$	$(q_1, 1, R)$	$(q_{end}, 1, R)$

- למשל, אם $\delta(q_0, 0) = (q_0, 0, R)$ זה אומר שאם קראנו בסרט את האות 0 ואנחנו במצב q_0 אז בתא הבא מימין אנחנו צריכים לכתוב 0 ולכן נשאר במצב q_0 ונזיז את הראש הקורא/כותב ימינה (R) לתא הבא בסרט.
- למשל, אם $\delta(q_0, \bar{b}) = (q_{end}, 0, R)$ זה אומר שאם קראנו בסרט את האות $\bar{b}$ ממצב q_0 אז נכתוב בתא הזה "0" ואז נעבור למצב מסיים q_{end}, ונזוז ימינה (R) לתא הבא. אנחנו לא נשארים (S) כיוון שלפי ההגדרה, המילה שחוזרת מהפונקציה היא רצף התווים משמאל לראש ולכן אנחנו צריכים לזוז צעד אחד ימינה לאחר שכתבנו את ה-"0" כדי שנוכל להחזיר גם אותו.

דוגמה 2 - בנייה למכונת טיורינג המחשבת פונקצייה

בנו מכונת טיורינג M שמחשבת את הפונקציה $f_M(w) = \varepsilon$.

<u>פתרון</u>: יהי מכונת טיורינג $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$.

לא משנה איזה קלט $w \in \Sigma^*$ הפונקציה f_M תקבל, היא תמיד תחזיר את המילה הריקה.

לכן, אנחנו לא צריכים לעבור על המילה של הקלט כי הפונקציה תמיד מחזירה ערך אחד ויחיד: ε.

אנחנו נקבל כקלט את $w \in \Sigma^*$ ונעבור ישר למצב סופי $q_{end} \in F$.

<u>הסבר</u>: אמרנו שלפי הגדרה, כש-M מחשבת פונקציה, היא צריכה להחזיר את כל רצף התאים שמשמאל לראש הכותב/קורא. בתחילת הריצה, הראש תמיד מתחיל כשהוא מצביע על התא הראשון, לכן אם לא נזיז אותו בכלל אז נחזיר את כל מה שמשמאלו, מכיוון שאין כלום משמאל אז זה אומר שנחזיר כלום כלומר ε כנדרש לפונקציה.

דוגמה 3 - בנייה למכונת טיורינג המחשבת פונקצייה

בנו מכונת טיורינג M שהולכת תמיד ימינה מבלי לעצור. מהי הפונקציה של המכונה שבניתם?

<u>פתרון</u>: יהי מכונת טיורינג $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$.

לא משנה איזה קלט $w \in \Sigma^*$ אנחנו מקבלים, תמיד צריך ללכת ימינה ולא לעצור.

אם המכונה בריצה אינסופית (לא עוצרת לעולם) אז קבוצת המצבים הסופיים $F = \emptyset$.

קבוצת המצבים Q מכילה רק מצב אחד $Q = \{q_0\}$.

ומכיוון שהמכונה תמיד זזה ימינה מבלי לעצור, פונקציית המעברים מוגדרת כלולאה עצמית על המצב q_0

כאשר ראש הסרט תמיד זז ימינה: $\delta(q_0, \sigma) = (q_0, \sigma, R)$.

הפונקציה $f_M(w)$ של המכונה M לא מוגדרת לכל קלט $w \in \Sigma^*$.

דוגמה 4 - בנייה למכונת טיורינג המחשבת פונקצייה

בנו מכונת טיורינג M המחשבת את הפונקציה $f(w) = 1^{|w|}$.

<u>פתרון</u>: יהי מכונת טיורינג $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$. הפונקציה f מקבלת קלט $w \in \Sigma^*$ וממירה את w למילה של אחדות באורך $|w|$. נגדיר קבוצת מצבים סופיים $F = \{q_{end}\}$ וקבוצת מצבים $Q = \{q_0, q_{end}\}$.

א"ב עבודה מוגדר בצורה הבאה: $\Gamma = \Sigma \cup \{1\}$.

פונקציית המעברים של המכונה מוגדרת באופן הבא:

- לכל $\sigma \in \Sigma$ מתקיים: $\delta(q_0, \sigma) = (q_0, 1, R)$.
- עבור $\bar{b} \in \Gamma$ מתקיים: $\delta(q_0, \bar{b}) = (q_{end}, \bar{b}, S)$.

דוגמה רצינית - בנייה למכונת טיורינג המחשבת פונקצייה

בנו מכונת טיורינג M המחשבת את $f_M(x) = x + 1$ כאשר x נתון בייצוג בינארי.

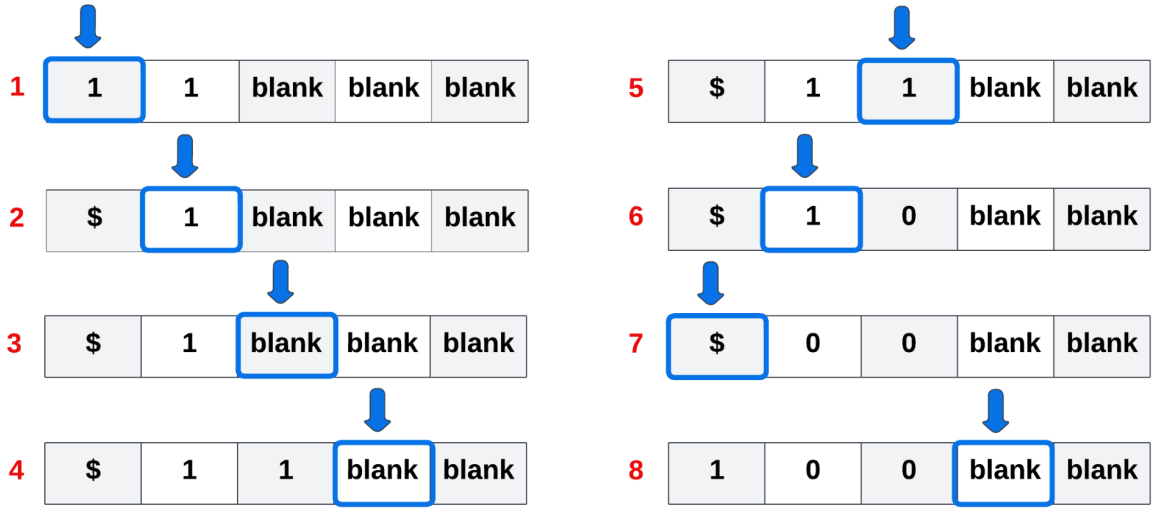
<u>רעיון הפתרון</u>:

1. נתחיל במצב q_0, כדי לטפל במצב של הרחבת המחרוזת (Carry) נדחוף סימון חדש $ בתא הראשון.
2. נבצע Shift Right ל-x אם קראנו 0 נעבור למצב q^0_{SR} ואם קראנו 1 נעבור למצב q^1_{SR}.
3. כשנסיים להזיז את x ימינה, נעבור למצב q_{add} שמטפל בפעולת החיבור + 1: נטייל שמאלה כל עוד התא מכיל את המספר 1, ונחליף כל תא כזה ב-0 עד כאשר בתא:
   a. נראה $ אז נחליף אותו ל-1 ואז נעבור למצב q_R שמזיז את הראש ימינה עד שנפגוש את ה-$\bar{b}$ הראשון כדי להוציא את המספר החדש כפלט.
   b. נראה 0 אז נשנה את התא הזה ל-1, אם נפגשנו ב-0 זה אומר שאנחנו לא מרחיבים את המחרוזת, כלומר ה-$ נשאר בתא הראשון וזה לא פלט תקין. לכן נרצה לעשות shift left:
      i. נעבור למצב q_1 שיקח אותנו ימינה עד ל-$\bar{b}$ הראשון שיעביר אותנו ל-$q^{\bar{b}}_{SL}$.
      ii. נזוז צעד אחד שמאלה (המספר הימני ביותר), אם הוא 0 נעבור למצב q^0_{SL} ואחרת נעבור למצב q^1_{SL} עד כאשר נפגוש את ה-$, נדרוס אותו במספר לפי המצב הנתון.
      iii. נעבור למצב q_R שמזיז את הראש ימינה עד שנפגוש את ה-$\bar{b}$ הראשון כדי להוציא את המספר החדש כפלט.

<u>פתרון</u>: נבנה טבלת מעברים שתמדל את הבעיה:

	0	1	♭	$
q_0	$q^0_{SR}, \$, R$	$q^1_{SR}, \$, R$	q_f	-
q^0_{SR}	$q^0_{SR}, 0, R$	$q^1_{SR}, 0, R$	$q_{add}, 0, S$	-
q^1_{SR}	$q^0_{SR}, 1, R$	$q^1_{SR}, 1, R$	$q_{add}, 1, S$	-
q_{add}	$q_1, 1, R$	$q_{add}, 0, L$	-	$q_R, 1, R$
q_1	$q_1, 0, R$	$q_1, 1, R$	$q^{♭}_{SL}, ♭, L$	-
$q^{♭}_{SL}$	$q^0_{SL}, ♭, L$	$q^1_{SL}, ♭, L$	-	-
q^0_{SL}	$q^0_{SL}, 0, L$	$q^1_{SL}, 0, L$		$q_R, 0, R$
q^1_{SL}	$q^0_{SL}, 1, L$	$q^1_{SL}, 1, L$		$q_R, 1, R$
q_R	$q_R, 0, R$	$q_R, 1, R$	$q_f, ♭, S$	-

דוגמת סימולציה:



קונפיגורציות

קונפיגורציה היא מעיין תמונת מצב במהלך ריצת המכונה - לפני או אחרי השלמה של פעולת חישוב ב-δ.
ניתן לתאר את ריצת מכונת טיורינג על ידי סדרת קונפיגורציות.
קונפיגורציה c היא שלשה $c = (\alpha, q, i)$ כאשר:

- $\alpha \in \Gamma^*$ הוא מחרוזת תוכן הסרט. (עד ה-$\bar{b}$ הראשון שהחל ממנו הכל רק $\bar{b}$, לא כולל).
- $q \in Q$ הוא המצב הנוכחי.
- $i \in \mathbb{N}$ הוא מיקום הראש הקורא כותב. (אינדקס).

לדוגמה: תוכן הסרט הוא $\alpha = 1001\bar{b}011$, המצב q ומיקום הראש הכותב $i = 5$ ולכן $\alpha_i = \alpha_5 = \bar{b}$.

תוכן הסרט α

בקונפיגורציה $c = (\alpha, q, i)$ תוכן הסרט $\alpha \in \Gamma^*$ הוא קצת מורכב, מכיוון שבקונפיגורציה הראשונה c_0
מדובר על הקלט של המכונה, אבל עם התקדמות המכונה, ייתכן ויהיה כתוב שם תוכן שונה לחלוטין.
כלומר, ייתכן שנרחיב את תוכן הסרט במהלך הריצה ולכן צריך שיטה שתקבע לנו את התוכן באופן "דינמי".
אז איך נקבע מה יהיה תוכן הסרט α? -

- נסמן ב-T את מספר הצעדים שמ"ט M ביצעה עד כה על הקלט x.
- נסמן ב-t את האינדקס של התא הימני ביותר בו ביקרנו בסרט.
- מכיוון שבכל פעם אנחנו זזים רק צעד אחד ימינה אז מתקיים $t \leq T$.
- לכן, תוכן הסרט α יכיל את ה-$max\{t, |x|\}$ התאים הראשונים של הסרט.

<u>*הסבר*</u>: מה שמעניין אותנו בסרט זה: או הקלט או התא הכי ימני שנפגשנו בו במהלך ריצת המכונה.
מכיוון שהסרט הוא אינסופי, אנחנו לא צריכים את כל ה-$\bar{b}$ האינסופיים איתנו ולכן הגדרנו את t, T כדי "לתפוס" את התאים הרלוונטים בלבד עבור תוכן הסרט שעלול להשתנות במהלך הריצה.
בנוסף, אורך תוכן הסרט הוא לא מספר קבוע כיוון שניתן לכתוב עליו במהלך הריצה ולהרחיב אותו.

סוגי קונפיגורציות

קונפיגורציה התחלתית היא $c_0 = (x, q_0, 1)$ כאשר x היא המילה אותה קיבלנו כקלט.

עבור u תוכן הסרט הנוכחי ו-i הוא מיקום הראש הנוכחי אז:

- **קונפיגורציה מקבלת** היא $c = (u, q_{accept}, i)$.
- **קונפיגורציה דוחה/מגעילה** היא $c = (u, q_{reject}, i)$.
- **קונפיגורציה עוצרת (סופית)** היא $c = (u, q_{accept}, i)$ או $c = (u, q_{reject}, i)$ או עם $q \in F$ כלשהו.

צעד חישוב

צעד חישוב במכונת טיורינג המתחיל ב- $c = (\alpha, q, i)$ יבוצע בצורה הבאה:
אם $\delta(q, \alpha_i) = (p, b, d)$ אז:

- נעבור ממצב q למצב p.
- האות α_i יוחלף באות $b \in \Gamma^*$.
- המיקום של הראש (האינדקס) יוחלף לפי ההזזה $d \in \{L, S, R\}$.

אם נזוז שמאלה $d = L$ אזי מעדכנים את המיקום לפי $max\{1\ ,\ i - 1\}$ (כיוון שלא ניתן לזוז שמאלה יותר מהתא הראשון של הסרט - אינדקס מספר 1, כי הסרט חסום משמאל).

רצף צעדי חישוב

רצף צעדי חישוב הם סדרת קונפיגורציות שעוברים עליהם במהלך ריצת מכונת טיורינג.
<u>הגדרה</u>: חישוב של מ"ט M על הקלט x היא סדרת קונפיגורציות $c_0, c_1, ..., c_{n-1}$ המקיימת:

1. נסמן ב-c_0 את הקונפיגורציה ההתחלתית.
2. לכל $i > 0$ מתקיים: $c_{i-1} \vdash_M c_i$.
3. אם קיימת קונפיגורציה סופית $c = (\alpha, q, i)$ אז היא האחרונה בסדרה והפלט של M על הקלט x מוגדרת להיות $\alpha[1 \dots i-1]$. (כי אמרנו שהפלט מוגדר ברצף התאים משמאל לראש הקורא/כותב).
4. אם הסדרה לא מכילה קונפיגורציה סופית, אזי הסדרה היא אינסופית והפלט של M על הקלט x לא מוגדר. במקרה זה נסמן כי הפלט הוא $f_M(x) = \perp$.

תרגיל סדרת קונפיגורציה

עבור המכונה מדוגמה 4 של בנייה למכונת טיורינג המחשבת פונקציה $f(w) = 1^{|w|}$.
תארו את ריצתה על הקלט 1101011 על ידי סדרת קונפיגורציות:

פתרון:

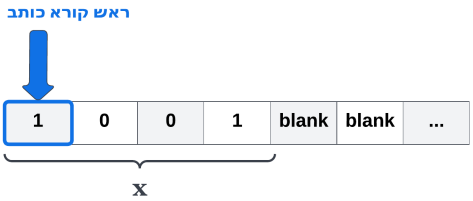
$$(1101011,\ q_0,\ 1) \vdash_M (1101011,\ q_0,\ 2) \vdash_M (1101011,\ q_0,\ 3)$$
$$\vdash_M (1111011,\ q_0,\ 4) \vdash_M (1111011,\ q_0,\ 5) \vdash_M (1111111,\ q_0,\ 6)$$
$$\vdash_M (1111111,\ q_0,\ 7) \vdash_M (1111111,\ q_0,\ 8) \vdash_M (1111111,\ q_{end},\ 8)$$

דוגמת סימולציה של מכונת טיורינג עם קונפיגורציות

נניח שאנחנו במכונה M המחשבת את הפונקציה $f_M(x) = 0 \cdot x$ כאשר $x = 1001$.

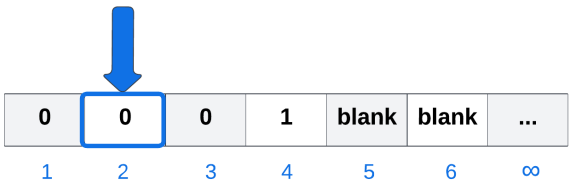
נתונה טבלת המעברים של הפונקציה:

$Q \backslash F$ / Γ	0	1	$\bar{b}$
q_0	$(q_0, 0, R)$	$(q_1, 0, R)$	$(q_{end},\ 0,\ R)$
q_1	$(q_0, 1, R)$	$(q_1, 1, R)$	$(q_{end},\ 1,\ R)$

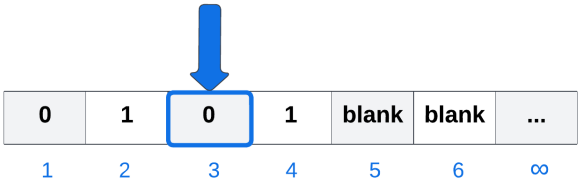


הקונפיגורציה ההתחלתית היא $c_0 = (1001, q_0, 1)$.

כעת מכיוון ש: $\alpha[1] = 1$ אז נעבור ל-q_1 אבל כיוון שאנחנו צריכים לשרשר "0" בתחילת המחרוזת אז נדרוס את התא הראשון $\alpha[1] = 0$ לכן: $c_1 = (0001, q_1, 2)$
(דרסנו את ה-"1" בהתחלה ושמנו במקומו את "0").

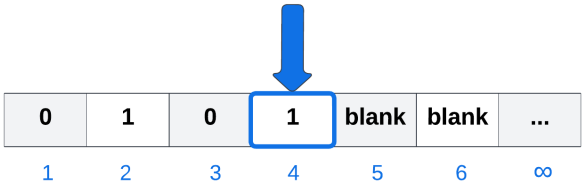


מכיוון שבתא $\alpha[2] = 0$ אז נעבור למצב q_0 אך לפני שנעבור, בגלל שאנחנו נמצאים במצב q_1 אז נדרוס $\alpha[2] = 1$ ואז נעבור למצב q_0. לכן: $c_2 = (0101,\ q_0,\ 3)$.

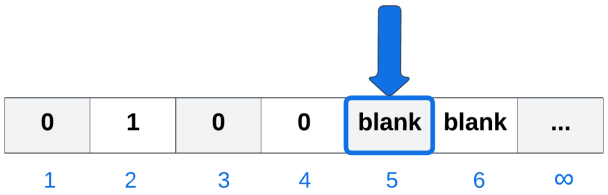


כעת בתא $\alpha[3] = 0$ ומכיוון שאנחנו במצב q_0 אז נשאר באותו המצב, נכתוב $\alpha[3] = 0$ ונזוז צעד ימינה.

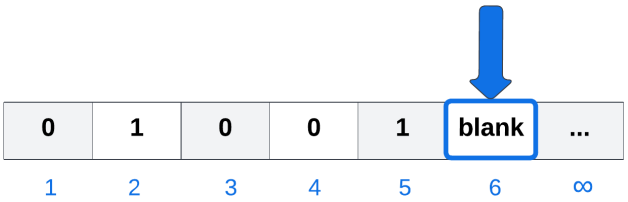
לכן $c_3 = (0101, q_0, 4)$.



מכיוון שבתא $\alpha[4] = 1$ אז נעבור למצב q_1 אך לפני שנעבור, בגלל שאנחנו נמצאים במצב q_0 אז נדרוס $\alpha[4] = 0$ ואז נעבור למצב q_1. לכן: $c_4 = (0100, q_1, 5)$.



כעת, מכיוון ש: $\alpha[5] = \bar{b}$ אז לפי טבלת המעברים, אם אנחנו במצב q_1 אז נדרוס $\alpha[5] = 1$ ואז נעבור למצב הסופי q_{end}. כלומר: $c_6 = (01001, q_{end}, 6)$.



שקילות מודלים

לכל **מודל** A של מכונת חישוב, נגדיר את F_A להיות קבוצת כל הפונקציות הניתנות לחישוב על ידי מכונה כלשהי מהמודל A. אז בהינתן שני מודלים שונים A, B נאמר שם שקולים אם"ם $F_A = F_B$.

- **מסקנה**: כדי להוכיח ששני מודלים שקולים, צריך להוכיח שהקבוצות של הפונקציות המתאימות לשני המודלים שוות. (בהכלה דו כיוונית).

דוגמה: תהי שפה $P = \{p \in \{0,1\}^* \mid p \text{ is binary representation of a prime number}\}$.
למדנו כי $NFA \equiv DFA$ אבל לא שקול ל-PDA (אוטומט מחסנית).
למדנו כי PDA שקול לדח"ה (כלומר לשפות חסרות הקשר), אבל מתברר שדח"ה לא שקול למכונת טיורינג.
באוטומטים למדנו שהשפה P היא לא ח"ה אבל מכונת טיורינג יכולה לעשות את זה כי היא שקולה למחשב מודרני ובמחשב קיימים אלגוריתמים שיודעים איך לפענח את השפה P (נחלק בכל הראשוניים עד השורש).

שקילות מכונות טיורינג

שתי מכונות טיורינג M_1, M_2 יקראו שקולות אם לכל קלט $x \in \Sigma^*$ התוצאה של שתי המכונות זהה.

- אם $M_1(x)$ מוגדר אז גם $M_2(x)$ מוגדר ושווה ל-$M_1(x)$.
- אם $M_1(x)$ לא מוגדר אז גם $M_2(x)$ לא מוגדר.
- $f_{M_1}(x) = f_{M_2}(x)$. כלומר, אם הפלט של האחד לא מוגדר אז גם הפלט של השני לא מוגדר.

יכולים להיות קלטים שהפלט לא מוגדר עליהם כמו למשל "פונקציות לא מלאות".

דוגמה: מכונת טיורינג זריזה, שבה מותר גם לזוז שני צעדים לכל כיוון ולא רק אחד.
רעיון ההוכחה: כיוון אחד - אם יש לנו מ"ט רגילה אז היא גם זריזה פשוט לא השתמשה ב-2 ימינה או 2 שמאלה.
כיוון שני - אם יש לנו מ"ט זריזה, אפשר לבנות לה מ"ט רגילה עם עוד 2 מצבים חדשים q_L, q_R במסעי ε.

איך מוכיחים שקילות מודלים בתכל'ס?

⇐ כיוון ראשון: ניקח מכונה M_A (כללית) מהמודל הראשון A, ונמיר אותה למכונה שקולה M_B מהמודל השני B.
כלומר, נדאג שהפונקציה אותה המכונה החדשה מחשבת, זהה לפונקציה של המכונה הראשונה.
⇒ כיוון שני: אותו הדבר רק הפוך. (ניקח מכונה מהמודל השני ונמיר למכונה שקולה מהמודל הראשון).
הערה: בדרך כלל אחד הכיוונים יהיה קל להוכיח והשני ידרוש מעט עבודה.

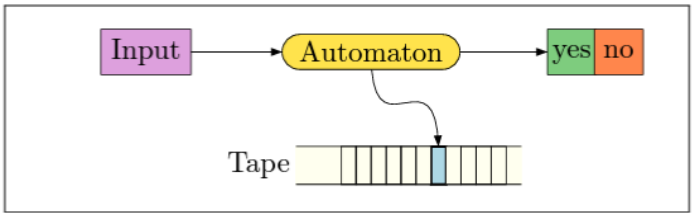
תזת צ'רץ'-טיורינג

התזה: כל מודל **כללי וסביר** שקול למכונת טיורינג.

- **כללי** = הכוונה למודל חזק לפחות כמו מכונת טיורינג.
- **סביר** = הכוונה למודל שלא מכיל רכיבים אינסופיים.

שאלה: אבל למכונת טיורינג יש רכיב אינסופי שהוא הסרט האינסופי.

תשובה: המשמעות מבחינתנו של הסרט האינסופי, היא רק שבמהלך הריצה של מכונת טיורינג לא יקרה מצב שחסר מקום אחסון. בפועל, כל כמות של תווים שנוכל לכתוב על גבי הסרט היא תמיד סופית.
כלומר, אם נתייחס לאורך המילה כקבוע, תמיד נשתמש באורך סרט סופי אבל לא חסום.

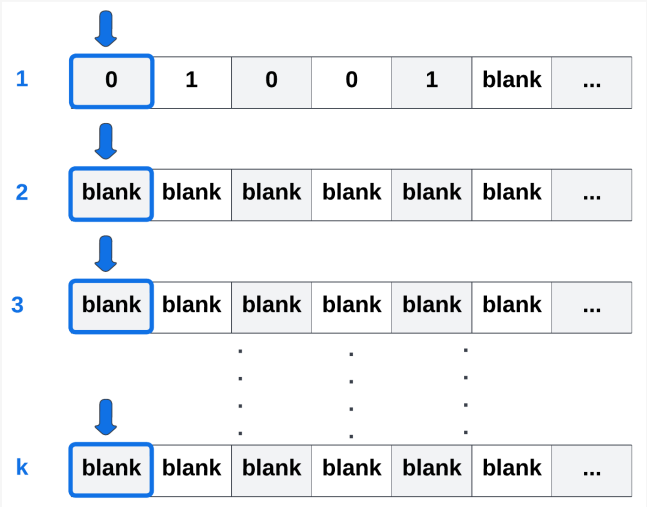


שינויים אפשריים למכונת טיורינג

מה אפשר לשנות במכונת טיורינג? במהלך הקורס נעשה שינויים במכונה שיגידו לנו האם המודל החדש חזק מדי או שהוא מודל ששקול לטיורינג רגיל.

1. אפשר להוסיף מספר קבוע של סרטים אינסופיים. (במ"ט רגילה יש סרט אחד).
2. אפשר לגרום לסרט להיות אינסופי לשני הכיוונים.
3. ניתן לעשות שינויים קלים בקבוצת הצעדים האפשריים.
4. אפשר להוסיף מספר קבוע של ראשים קוראים/כותבים.
5. אפשר להוסיף כל שינוי שלא כולל רכיב אינסופי.

מכונת טיורינג מרובת סרטים



עד עכשיו הגדרנו מודל יחיד של מכונת טיורינג.
נציג מכונה בעלת k סרטים (עבור $k \geq 1$).

- לכל סרט יש ראש קורא-כותב משלו.
- בתחילת הריצה, הקלט רשום בסרט מספר 1 ושאר הסרטים ריקים (מלאים ב-$\bar{b}$).
- פונקציית המעברים מוגדרת:
$\delta: Q' \times \Gamma^k \to Q \times \Gamma^k \times \{L,S,R\}^k$.
- הפלט מוגדר להיות תוכן הסרט הראשון בזמן העצירה.

הוכחת שקילות מ"ט רגילה עם מ"ט דו-סרטית

נסמן ב-F_M את קבוצת הפונקציות שניתנות לחישוב במ"ט רגילה (עם סרט יחיד).

נסמן ב-F_{M^2} את קבוצת הפונקציות שניתנות לחישוב במ"ט דו-סרטית (עם 2 סרטים).

נרצה להוכיח כי $F_{M^2} = F_M$.

⇐ <u>כיוון ראשון</u> $F_M \subseteq F_{M^2}$ - מ"ט רגילה היא בפרט מ"ט דו-סרטית רק שאינה מבצעת שימוש בסרט השני.

תהא $M = (Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta)$ מ"ט רגילה. נתאר את $M^2 = (Q, q_0, F, \Gamma, \Sigma, \bar{b}, \bar{\delta})$ מ"ט דו-סרטית.

אם $\delta(q, \sigma) = (q', \sigma', m)$ אז $\bar{\delta}(q, \sigma, \tau) = (q', \sigma', 0, m, S)$.

בעצם M^2 עושה אותו הדבר כמו M ופשוט לא משתמשת בסרט השני.

⇒ <u>כיוון שני</u> $F_M \subseteq F_{M^2}$ - תהא $M^2 = (Q_{M^2}, q_0, F_{M^2}, \Gamma_{M^2}, \Sigma_{M^2}, \bar{b}_{M^2}, \delta_{M^2})$ מ"ט בעלת 2 סרטים.

נראה שלמ"ט רגילה $M = (Q_M, q_0, F_M, \Gamma_M, \Sigma_M, \bar{b}_M, \delta_M)$ יש את אותו "כח" כמו מכונה M^2.

נתאר את M - מ"ט רגילה המחשבת את אותה הפונקציה: קבוצת המצבים $Q_M = Q_{M^2}$. הא"ב $\Sigma_M = \Sigma_{M^2}$.

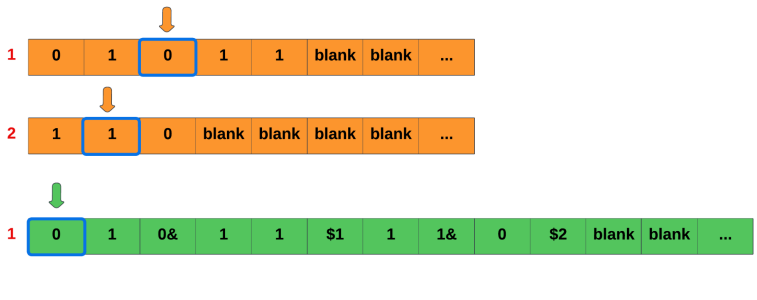
הא"ב עבודה מוגדרת כך: $\Gamma_M = \{0, 1, \bar{b}_{M^2}\} \cup \{\$_1, \$_2, 0\&, 1\&, \bar{b}\&\}$.

- תוכן הסרט ה-1 : $\alpha_1 \in \Gamma_{M^2}$ נכתב לסרט של M החל מה-$\bar{b}$ הראשון, כשבסופו נצרף את האות המיוחדת $\$_1$. אם הראש של הסרט ה-1 מצביע על האות $\sigma \in \{0, 1, \bar{b}\}$ אז בסרט של M נסמנו $\sigma\&$.
- תוכן הסרט ה-2 : $\alpha_2 \in \Gamma_{M^2}$ נכתב לסרט של M החל מה-$\bar{b}$ הראשון, כשבסופו נצרף את האות המיוחדת $\$_2$. אם הראש של הסרט ה-2 מצביע על האות $\sigma \in \{0, 1, \bar{b}\}$ אז בסרט של M נסמנו $\sigma\&$.

כאשר המכונה הדו-סרטית M^2 תבצע שינויים (הזזת הראש וכתיבה/קריאה) ב-2 הסרטים שלה, אנחנו נעדכן בהתאם גם בסרט היחיד של M באמצעות הסימונים המיוחדים שהוספנו לה.

את התיאור הפורמלי של השינויים האלה אנחנו לא נכתוב כי זה הרבה עבודה.

דוגמה ויזואלית (לא חלק מההוכחה):



נעתיק תחילה את תוכן הסרט הראשון של M^2 לתחילת הסרט של M. לאחר מכן נסמן עם $\$_1$ שאומר לנו "שכאן" נגמר תוכן הסרט הראשון. לאחר מכן נעתיק גם את תוכן הסרט השני ל-M וגם לו נוסיף בסוף את $\$_2$. הראש של הסרט הראשון של M^2 הצביע על תא מספר 3 ולכן נסמן בתא 3 של M את $0\&$. באותו אופן עבור סרט מספר 2.

הוכחת שקילות מ"ט רגילה עם מ"ט k-סרטית

נסמן ב-F_M את קבוצת הפונקציות שניתנות לחישוב במ"ט רגילה (עם סרט יחיד).

נסמן ב-F_{M^k} את קבוצת הפונקציות שניתנות לחישוב במ"ט k-סרטית (עם k סרטים).

נרצה להוכיח כי $F_{M^k} = F_M$.

לא נוכיח כאן באופן מלא כי זה דומה להוכחת השקילות בין מ"ט רגילה לבין מ"ט דו-סרטית שכבר הוכחנו.

⇐ <u>כיוון ראשון</u> - $F_M \subseteq F_{M^k}$ מ"ט רגילה היא בפרט מ"ט k-סרטית שמבצעת שימוש רק בסרט אחד בלבד.

ההמשך דומה כמו שעשינו עם הדו-סרטית.

⇒ <u>כיוון שני</u> - $F_{M^k} \subseteq F_M$ תהא $M^k = (Q_{M^k}, q_0, F_{M^k}, \Gamma_{M^k}, \Sigma_{M^k}, \bar{b}_{M^k}, \delta_{M^k})$ מ"ט בעלת k סרטים.

עובד באותו האופן כמו שהוכחנו עבור הדו-סרטית עם שינוי קל שמעביר אותנו לכל k כללי וקבוע:

לכל סרט $i \in [1, k]$ (בסדר עולה) של המכונה M^k נפעל בצורה הבאה:

- תוכן הסרט ה-i: $\alpha_i \in \Gamma_{M^k}$ נכתב לסרט של M החל מה-$\bar{b}$ הראשון, כשבסופו נצרף את האות המיוחדת $\$_i$. אם הראש של הסרט ה-$i$ מצביע על האות $\sigma \in \{0, 1, \bar{b}\}$ אז בסרט של M נסמנו σ& בהתאם.

כאשר המכונה ה-k סרטית M^k תבצע שינויים (הזזת הראש וכתיבה/קריאה) ב-k הסרטים שלה, אנחנו נעדכן בהתאם גם בסרט היחיד של M באמצעות הסימונים המיוחדים שהוספנו לה.

את התיאור הפורמלי של השינויים האלה אנחנו לא נכתוב כי זה הרבה עבודה.

דוגמה - מ"ט דו-סרטית

נתאר מילולית מ"ט דו-סרטית שמחשבת את הפונקציה $f(x) = xx$.

נתונים 2 סרטים, לפי הגדרת המכונה, בראשון קיים תוכן x והשני ריק (כולו blank).

אנו יודעים גם שהסרט הראשון במכונה כזאת תכיל את הפלט שלנו ולכן נרצה לכתוב לתוכה את התשובה.

1. כתוב את הסימן המיוחד $ בתחילת הסרט השני.
2. עבור על x בסרט 1 והעתק אותו במקביל לסרט 2.
3. החזר את הראש בסרט 2 לתחילת ה-x באמצעות הסימן המיוחד $.
4. העתק את x מסרט 2 להמשך סרט 1.
5. החזר את כל התוכן שנמצא משמאל לראש של סרט 1.

מכונת טיורינג עם קבוצת מצבים אינסופית

נגדיר **מודל אינסופי** M * הזהה למכונת טיורינג הרגילה אבל שקבוצת המצבים יכולה להיות אינסופית.

טענה: המודל לא שקול למכונת טיורינג.

הסבר: כל פונקציה הינה רשימה אינסופית של מחרוזות. אזי, בהינתן פונקציה כלשהי, ניתן לשמור את כל המחרוזות המתאימות באינסוף המצבים הנתונים לנו. בנוסף אמרנו שלפי התזה של טיורינג, כל הרכיבים צריכים להיות סופיים, אבל הרכיב "קבוצת המצבים" היא אינסופית ולכן מראש היא לא שקולה למ"ט רגילה.

דוגמה למודל אינסופי: קבוצה המצבים של M * היא $Q = \{q_\varepsilon, q_0, q_1, q_{00}, q_{01}, q_{10}, q_{11}, \ldots\}$.

כל מצב כזה "זוכר" את הפלט שמתאים לו, כאשר נקרא את המילה נוכל לעבור בין המצבים לפי המילה שקראנו. ולבסוף, כאשר נגיע לרווח הראשון (blank) נדע שסיימנו לקרוא את המילה ונחזיר את הפלט שהמצב שהגענו אליו זוכר. כלומר, נחזור לתחילת הסרט ונכתוב את הפלט המתאים.

מכונת טיורינג זריזה

מכונה זו מוגדרת בדומה למכונת טיורינג רגילה, רק שאפשר לבצע בה גם שני צעדים ימינה ושני צעדים שמאלה. פונקציית המעברים שלה מוגדרת כך: $\delta: (Q\backslash F) \times \Gamma \to Q \times \Gamma \times \{LL, L, S, R, RR\}$.

טענה: מודל מכונת טיורינג זריזה שקול למודל מכונת טיורינג רגילה.

רעיון ההוכחה: על מנת להוכיח שקילות מודלים, ניקח מכונה הפועלת במודל אחד ונראה מכונה שקולה לה במודל השני, ולהיפך. (כמו שעשינו בשקילות מודלים באוטומטים).

הוכחה:

⇐ כיוון ראשון: תהי M מכונת טיורינג רגילה. נבנה מכונת טיורינג זריזה M' שתחשב את אותה הפונקציה ש-M מחשבת. נשים לב ש-M מתאימה למודל של מ"ט זריזה, פשוט לא משתמשת באפשרויות RR ו-LL, לכן נוכל לבחור $M' = M$.

⇒ כיוון שני: תהי M' מכונת טיורינג זריזה, נבנה מ"ט M רגילה שתחשב את אותה הפונקציה ש-M' מחשבת, כלומר שהיא תמדל את התוסף של LL, RR. נגדיר אותה כך: $M = (Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta)$.

את רוב הרכיבים אין צורך לשנות מלבד Q, δ.

המוטיבציה שלנו היא "לזכור" כשהמכונה הזריזה M' רוצה לזוז שני צעדים באותו כיוון, ותיישם זאת.

עבור קבוצת המצבים, על כל מצב $q_i \in Q'$ במכונה הזריזה נוסיף לו 2 מצבים חדשים q_{iL}, q_{iR} במכונה הרגילה. כלומר נגדיר כך:

$$Q := Q' \cup Q_L \cup Q_R$$
$$Q_L := \{q_{iL} \mid q_i \in Q'\}$$
$$Q_R := \{q_{iR} \mid q_i \in Q'\}$$

עבור פונקציית המעברים δ:

- בכל פעם שפונקציית המעברים של הזריזה δ' משתמשת ב-L, R, S אז נגדיר את הפונקציה:

$$\delta(q,a) := \delta'(q,a)$$

- בכל פעם שפונקציית המעברים של הזריזה δ' משתמשת ב-LL, RR אז נגדיר את הפונקציה:

$$\delta'(q,a) = (q_i, b_i, LL) \rightarrow \delta(q_i,a) = (q_{iL}, b_i, L)$$

$$\delta'(q,a) = (q_i, b_i, RR) \rightarrow \delta(q_i,a) = (q_{iR}, b_i, R)$$

וכך "נזכור" את ההזזה הנוספת שיש לנו להשלים.

ההשלמה תעשה כך: לכל q_i ולכל a נבצע:

$$\delta(q_{iL}, a) = (q_i, a, L)$$

$$\delta(q_{iR}, a) = (q_i, a, R)$$

תרגיל בית - שקילות מכונות

תהי מ"ט M' בעלת הכיוונים: $\{S,\ Right,\ Reset\}$ כאשר $Reset$ מחזירה את הראש לתא הראשון (1).

האם M' שקולה למ"ט M רגילה?

<u>הכוונה</u>: כדי להפריך, מספיקה שפה אחת שניתן להכריע בעזרת אחת מהן ולא בעזרת השנייה.

כדי להוכיח, יש להראות איך מבצעים $Left$ במודל החדש.

<u>הוכחה דו כיוונית</u>:

⇐ <u>כיוון ראשון</u> $F_{M'} \subseteq F_M$: תהי M מכונת טיורינג רגילה. נבנה מכונת טיורינג M' שתחשב את אותה הפונקציה ש-M מחשבת. קבוצת המצבים $Q_{M'} = Q_M \cup \{q_{Left}\}$, קבוצת המצבים הסופיים $F_M = F_{M'}$ וגם $\Sigma_M = \Sigma_{M'}$.

א"ב עבודה יתואר באופן הבא: $\Gamma_{M'} = \{0, 1, \bar{b}\} \cup \{0\&,\ 1\&,\ \bar{b}\&\}$.

הראש הכותב של הסרט של M שמצביע על האות $\sigma \in \Gamma_M$ יוחלף באותו מקום בסרט של M' עם הסימון $\sigma\&$.

- כאשר M מפעיל צעד $Left$ אז נכתוב בתא שלאחר ההזזה את $\sigma\&$. נעבור ב- M' למצב q_{Left} שמבצע $Reset$ לראש ואז יבצע חיפוש מימין $Right$ עד שיפגוש את התא שמסומן ב-$\sigma\&$ ואז נעבור ממצב q_{Left} לאותו מצב ש M נמצא בו.
- כאשר M מפעיל צעד $Right$ או S אז M' יפעל באופן שווה כי הפעולות האלו קיימות גם בו.

⇒ <u>כיוון שני</u> $F_{M'} \subseteq F_M$: תהי M' מ"ט החדשה. נבנה מ"ט M רגילה שתחשב את אותה הפונקציה ש-M' מחשבת. כל הרכיבים שווים מלבד פונקציית המעברים וקבוצת המצבים $Q_M = Q_{M'} \cup \{q_{Reset}\}$.

- כאשר M' מפעיל צעד $Reset$, נעבור ב-M למצב q_{Reset} שמזיז את הראש עם צעדי $Left$ עד שנגיע לתחילת הסרט. כשנגיע להתחלה, נעבור ממצב q_{Reset} לאותו המצב ש-M' נמצא בו.
- כאשר M' מפעיל צעד $S,\ Right$ אז נפעל באופן בשווה גם במודל M.

השוואה בין מ"ט לבין אוטומטים פשוטים

ראינו שמ"ט הוא מודל חזק יותר משפות ח"ה ולכן גם משפות רגולריות.

בקורס אוטומטים למדנו כי השפה $L = \{0^n1^n \mid n > 0\}$ לא רגולרית ולכן לא ניתן לבנות לה אוטומט $DFA,\ NFA$ וכו'... בנוסף למדנו שאפשר לבנות לשפה הזאת אוטומט מחסנית ששקול לדח"ה (שפות ח"ה).
אם כך, בהכרח אפשר לבנות מודל ממ"ט שמקבל את השפה הזאת כי מ"ט חזק יותר משפות ח"ה.
נתאר מ"ט המכריעה את השפה L, אנחנו נתאר כמקובל במבחנים של הקורס: באמצעות פסואדו-קוד.
למה פסאודו-קוד? - כי הוא ניתן לחישוב על ידי מחשב ⇐ ניתן לחישוב על ידי מכונת טיורינג.

פסאודו-קוד: מכונת טיורינג המכריעה את $L = \{0^n1^n \mid n > 0\}$.

1	בדוק כי המילה מהצורה 0^n1^m כאשר $n \neq m$, אחרת - דחה.
2	סמן את האות הראשונה של המילה ב-x.
3	הזז את הראש ימינה עד שתגיע ל-1 הראשון וסמן אותו ב-y - אם לא נמצא "1" כזה, דחה.
4	לולאה: חזור שמאלה עד לתא שמימין ל-x הימני ביותר ותקבע:
4.1	אם התא הזה מכיל את המספר 0 אז סמנו ב-x וחזור לשלב 3.
4.2	אם התא הזה מסומן ב-y אז זוז ימינה עד לסוף המילה.
4.2.1	אם נפגשת ב-"1" במהלך הריצה ימינה עד לסוף המילה - דחה.
5	קבל.

- בשלב 4.1 - אם סימנו את 0 ב-x ועברנו לשלב 3 ולא מצאנו "1" מתאים עבורו זה אז $|1| < |0|$ ונדחה.
- בשלב 4.2.1 - אם התא הזה מסומן ב-y זה אומר שסיימנו לסמן את כל האפסים ב-x ולכן נותר לנו רק לבדוק ש: $|1| = |0|$... אם במהלך הריצה ימינה לסוף המילה נפגשנו ב-1 אז $|1| > |0|$ ונדחה.

<u>דוגמה ויזואלית</u>:

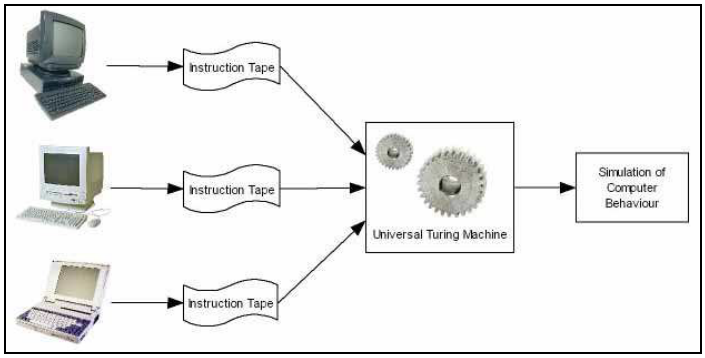
2	סמן את האות הראשונה של המילה ב-x.	**x001111bb**
3	הזז את הראש ימינה עד שתגיע ל-1 הראשון וסמן אותו ב-y - אם לא נמצא "1" כזה, דחה.	**x00y111bb**
4	לולאה: חזור שמאלה עד לתא שמימין ל-x הימני ביותר ותקבע:	**x00y111bb**
4.1	אם התא הזה מכיל את המספר 0 אז סמנו ב-x וחזור לשלב 3.	**xx0y111bb**
3	הזז את הראש ימינה עד שתגיע ל-1 הראשון וסמן אותו ב-y - אם לא נמצא "1" כזה, דחה.	**xx0yy11bb**
4	לולאה: חזור שמאלה עד לתא שמימין ל-x הימני ביותר ותקבע:	**xx0yy11bb**
4.1	אם התא הזה מכיל את המספר 0 אז סמנו ב-x וחזור לשלב 3.	**xxxyy11bb**
3	הזז את הראש ימינה עד שתגיע ל-1 הראשון וסמן אותו ב-y - אם לא נמצא "1" כזה, דחה.	**xxxyyy1bb**
4	לולאה: חזור שמאלה עד לתא שמימין ל-x הימני ביותר ותקבע:	**xxxyy1bb**
4.2	אם התא הזה מסומן ב-y אז זוז ימינה עד לסוף המילה.	
4.2.1	אם נפגשת ב-"1" במהלך הריצה ימינה עד לסוף המילה - דחה.	**xxxyyy1bb**
5	קבל	

מכונת טיורינג אוניברסלית

קידודים

קלט למכונת טיורינג הוא מחרוזת של סימנים $x \in \Sigma^*$. נרצה אלגוריתמים שיעבדו על מכונות טיורינג, גרפים, מטריצות, פולינומים ועוד ולכן נרצה לבחור קידוד $< X >$ לאובייקט X.

מכונת טיורינג אוניברסלית



מכונת טיורינג אוניברסלית היא המחשב המודרני של היום. עד עכשיו, חשבנו על מכונת טיורינג M כעל ייצוג של אלגוריתם - במחשב "אמיתי", המקבילה שלה היא **תוכנה**. מכונת טיורינג אוניברסלית U היא, אם כן, גם כן תוכנה - אבל תוכנה מסוג מיוחד: "**מערכת הפעלה**". הקלט של מכונת טיורינג אוניברסלית U הוא קידוד של מכונת טיורינג כלשהו, M, וקלט כלשהו, x. מה שהמכונה האוניברסלית U עושה הוא ל"הריץ" את M על x (כלומר, לבצע סימולציה או "סימלוץ" של M על x ולענות כמוהו).

נרצה לבנות מכונת טיורינג כללית שתוכל לבצע את המשימה של כל מכונת טיורינג אחרת.
כלומר, יהיו לנו כל מיני מכונות M שמריצות אלגוריתמים, אנחנו נרצה מכונה כללית U שתפעיל אותם.
מכונת טיורינג אוניברסלית U צריכה לקבל קידוד של מכונה M ואת הקלט $x \in \Sigma^*$ עבור M ותחזיר $M(x)$.
כלומר, נרצה שיתקיים:

$$U(< M, x >) = M(x)$$

על מנת לייצג מ"ט באופן נוח, נרצה לקודד אותה בצורת מחרוזת מעל הא"ב $\{0, 1\}$.
דרישות מקידוד $< M >$ של מכונת טיורינג:

1. יהיה ניתן לשחזר את המכונה M בהינתן הקידוד שלה $< M >$.
2. יהיה ניתן "להריץ" או "לסמלץ" את המכונה בהינתן הקידוד שלה $< M >$ על הקלט שלה $< x >$.

מונח	מקבילה בעולם שלנו
מכונת טיורינג אוניברסלית - U	מחשב מודרני / מערכת ההפעלה
מכונת טיורינג רגילה - M	אלגוריתם ספציפי במחשב / תוכנה

קידוד של מכונת טיורינג

אמרנו שמ"ט אוניברסלית U מקבלת קידוד של מ"ט M. כעת נראה שניתן לקודד מ"ט M למחרוזת.

- תהי שביעייה: $M = \left(Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ F = \{q_{accept}, q_{reject}\},\ \bar{b}\right)$.
- נניח בלי הגבלת הכלליות שמ"ט M היא מהצורה הבאה:

$$Q = \{q_1,\ q_2, \ldots, q_m\}\ ,\quad \Sigma = \{a_1,\ a_2, \ldots, a_k\}\ ,\quad \Gamma = \{b_1,\ b_2, \ldots, b_s\}$$

כאשר $q_0, q_{accept}, q_{reject}$ הם $q_1,\ q_2,\ q_3$ בהתאם.

כאשר $\bar{b}$ הם $a_1, a_2, \ldots, a_k, b_{k+1}$ בהתאם $b_1,\ b_2, \ldots, b_k$.

- נקודד: $L = 1\ ,\quad R = 2\ ,\quad S = 3$.
- לכל $\delta(q, a) = (p,\ b,\ LRS)$ נקודד בצורה הבאה: $< \delta(q,\ a) > \ = 1^p 0 1^b 123$.
- נפריד כל שני מעברי δ שונים על ידי 00.

לכן, הקידוד של המכונה M מתואר באופן הבא:

$$< M >= 1^{|Q|}01^{|\Gamma|}01^{|\Sigma|}0 < \delta(q_1, b_1) > 00 \ldots 00 < \delta(q_m,\ b_s) > 00$$

קידוד הקלט x

במכונת טיורינג אוניברסלית נרצה שיתקיים $U(< M, x >) \ = \ M(x)$.

עד עכשיו, קידדנו את המכונה M, אבל צריך בנוסף לקודד גם את x.

נשים לב שייתכן ש: $\Sigma_U \neq \Sigma_M$ ואז הקלט של M מורכב מתווים שזרים למכונה U וזה בעיה, איך U תתמודד?

אז את ה- $x \in \Sigma_M^*$ צריך לקודד בצורה כזו שגם U תוכל להריץ, לכן x צריך להיות בא"ב Σ_U.

כלומר, הקלט יהיה בדרך ככל בצורה הבאה: $< x >< M >$ ולא $< M > x <$.

הרי אמרנו שיהיה ניתן לשחזר את המכונה M בהינתן הקידוד שלה $< M >$, לכן נרצה שהמכונה U תדע לקרוא את המחרוזת x ולכן נקודד אותה בשביל ש-U תדע לקרוא אותה.

התאמת מכונות להרצה במ"ט אוניברסלית

צריך עדיין להוכיח שקיימת מכונה אוניברסלית U שיכולה לשחזר את הפעולה של כל מכונה שנתונה לה בתור קידוד. כלומר המכונה U יכולה "לחקות" את ריצת M על x:

1. המכונה U תשמור את הקונפיגורציה ההתחלתית של M בסרט נוסף (או בסוף הקלט של $< M >$). ראינו שמ"ט עם k סרטים שקולה למכונה רגילה עם סרט אחד, לכן אפשר לעבוד עם סרט נוסף.
2. המכונה U תשמור סימן מיוחד שמציין את מיקום הראש הקורא/כותב.
3. בכל שלב, המכונה U תעדכן את הקונפיגורציה הנוכחית של M, שנמצאת בסרט השני לפי ההוראות הנתונות בפונקציית המעברים δ שנמצאות כבר בקידוד שהכנו ל-M.
4. כאשר המכונה U תזהה מצב סופי של M (M עצרה) אז היא גם תעצור, ותחזיר את הפלט של M.

טיפול בקידוד לא תקין

מה קורה אם $< M >$ לא תקין? - יש שתי גישות עיקריות:

1. יחסית קל לבדוק בתחילת האלגוריתם אם הקלט תקין, לכן אפשר להניח שהקלט תקין.
2. לכל קידוד של מכונה שאיננו תקין, נתייחס בתור המכונה M_{STAM} העוברת מיידית למצב q_{accept}.

כלומר, כל מחרוזת בינארית שאיננה ניתנת לפירוש כקידוד של $< M >$, נבין אותה כקידוד של מכונת טיורינג M_{STAM} שעוצרת מיד (עוברת למצב q_{reject}).

ריצת מ"ט אוניברסלית

כיצד המכונה U תרוץ? - אנחנו נבנה מ"ט המממשת את U.

להלן סקיצה של מימוש מסוים, וזה יהיה "המכונה האוניברסלית" שעוד נפגוש המון בקורס:

- המכונה האוניברסלית M_U על הקלט $< M, x >$:
  - אם הקידוד $< x >$ לא חוקי אז נבצע לולאה אינסופית (נתקע במצב מסוים שהוא לא סופי).
  - נכתוב על הסרט השני את הקונפיגורציה $(< M >, < c >)$ כאשר $< c >$ היא c_0 של M.
  - כל עוד c לא קונפיגורציה סופית, נחשב את $NEXT(< M >, < c >)$ ונכתוב אותה בתור הקונפיגורציה הנוכחית (עדכון של c הקודמת). (ראינו שהעדכון נקבע לפי δ של M).
  - אם c קונפיגורציה סופית, אז נפלוט את קידוד הפלט המיוצג על ידי c.
- המכונה M_U מבצעת סימולציה צעד אחרי צעד של ריצת M על הקלט x.

כלומר, בכל שלב נפעיל את δ ונעדכן בהתאם את הקונפיגורציה - ככה עד שנגיע לקונפ' סופית.

המחלקות R ו-RE

תזכורת - מ"ט לקבלת שפות

<u>תזכורת</u>: שפה L הינה תת קבוצה סופית או אינסופית של Σ^*.

<u>הגדרה</u>: מכונת טיורינג לקבלת שפות הינה מכונת טיורינג רגילה שמקיימת $F = \{q_{acc}, q_{rej}\}$.

- המצב q_{acc} הינו מצב סופי שמשמעותו היא שהמכונה **מקבלת** את הקלט שלה.
- המצב q_{rej} הינו מצב סופי שמשמעותו היא שהמכונה **דוחה** את הקלט שלה.

<u>הערה</u>: גם למכונה לקבלת שפות יש פלט כמו שהוגדר במודל המקורי, אבל לרוב לא נתייחס אליו.

שפה של מכונה

בהינתן מכונה M עם הקלט שלה $x \in \Sigma^*$ אז:

- אם המכונה M בריצתה על x נעצרת על מצב q_{acc}, נאמר ש-M **מקבלת** את x ונסמן ב- $M(x) = 1$.
- אם המכונה M בריצתה על x נעצרת על מצב q_{rej}, נאמר ש-M **דוחה** את x ונסמן ב- $M(x) = 0$.
- אם המכונה לא עצרה - הפלט **לא מוגדר**.

<u>הגדרה</u>: שפת המכונה $L(M)$ הינה קבוצת כל המילים אשר המכונה M מקבלת אותם $(M(x) = 1)$.

<u>הגדרה</u>: השפה המשלימה $\overline{L(M)} = \Sigma^* \setminus L(M)$ היא קבוצת כל המילים ש-M דוחה **או** לא עוצרת.

ההבדל בין קבלת שפות להכרעת שפות

<u>הגדרה 1</u>: בהינתן שפה $L \subseteq \Sigma^*$, נאמר שמכונה M_L **מקבלת (מזהה)** את השפה L אם מתקיים:

$$x \in L \leftrightarrow M(x) = 1$$

<u>נקודה חשובה להגדרה 1</u>: אם $x \notin L$ אז $M(x) = 0$ או $M(x)$ לא עוצרת לעולם.

<u>הגדרה 2</u>: בהינתן שפה $L \subseteq \Sigma^*$, נאמר שמכונה M_L **מכריעה** את השפה L אם מתקיים:

$$x \in L \leftrightarrow M(x) = 1$$

ובנוסף, M_L **תמיד** עוצרת.

<u>נקודה חשובה להגדרה 2</u>: המכונה תמיד תעצור עבור כל מילה $x \notin L$ ותחזיר $M(x) = 0$.

הוכחת נכונות של שוויון מהצורה $L(M) = L$

בהינתן שפה L, שבנינו לה מכונה M, נצטרך להוכיח $L(M) = L$ (כלומר, ששפת המכונה היא אכן השפה L).
בתרגילים שנרצה להוכיח $L(M) = L$, אנחנו נוכיח את הטענה הבאה (תלוי אם היא מקבלת או מכריעה):

<u>טענה</u>: $L(M) = L$ אם לכל $x \in \Sigma^*$ מתקיים $x \in L \leftrightarrow x \in L(M)$.

- מבנה ההוכחה עבור מכונה **מקבלת**:
  - כל מילה $x \in L$ נרצה להראות את הפלט: $M(x) = 1$ כלומר $x \in L(M)$.
  - כל מילה $x \notin L$ נרצה להראות ש: $M(x) = 0$ או ש-$M(x)$ לא מוגדר כלומר $x \notin L(M)$.
- מבנה ההוכחה עבור מכונה **מכריעה**:
  - כל מילה $x \in L$ נרצה להראות את הפלט: $M(x) = 1$ כלומר $x \in L(M)$.
  - כל מילה $x \notin L$ נרצה להראות את הפלט: $M(x) = 0$ כלומר $x \notin L(M)$.

המחלקות R ו-RE

<u>הגדרה</u>: מחלקה R היא קבוצה שמכילה את כל השפות שקיימות עבורן מכונת טיורינג **המכריעה** אותן.

$$R = \{\forall L \subseteq \Sigma^* \mid \exists\ turing\ machine\ M\ s.t.\ L(M) = L \ \wedge\ M\ stops\ on\ every\ input\}$$

<u>הגדרה</u>: מחלקה RE היא קבוצה שמכילה את כל השפות שקיימות עבורן מכונת טיורינג **המקבלת** אותן.

$$RE = \{\forall L \subseteq \Sigma^* \mid \exists\ turing\ machine\ M\ s.t.\ L(M) = L\}$$

- למכונה מכריעה יש יותר כוח.
- הסימון R אומר $Recursive$.
- בהמשך נראה כי $R \subset RE$.
- הסימון RE אומר $Recursively\ Enumerable$.

דוגמאות לשפות ב-R

- השפות הטריוויאליות: $\emptyset$, Σ^*.
- השפות הרגולריות. - אוטומט סופי דטרמיניסטי DFA הוא מקרה פרטי של מכונת טיורינג (עם ראש קורא/כותב שעובר פעם אחת בדיוק על הקלט).
- שפות של בעיות מוכרות שידוע לנו על אלגוריתם שפותר אותן (תמיד) כמו למשל:
  - $L = \{x \mid x\ is\ a\ prime\ number\}$
  - $L = \{< G > \mid G\ is\ a\ connected\ graph\}$
  - $L = \{< G > \mid G\ is\ a\ graph\ that\ has\ an\ Hamilton\ path\}$

תכונות של המחלקות R, RE

תכונה 1 - $R \subseteq RE$: מ"ט המכריעה שפה, בהכרח גם מקבלת אותה.

תכונה 2 - המחלקה R סגורה למשלים: אם $L \in R$ אז $\bar{L} \in R$.

הוכחה 2 - תהי $L \in R$, לפי הגדרת R, קיים מכונה M המכריעה את השפה L.

נבנה מכונה $\overline{M}$ שמכריעה את השפה $\bar{L}$: המכונה $\overline{M}$ זהה ל-M למעט החלפה בין המצב המקבל והמצב הדוחה.

נראה נכונות עבור השפה המשלימה:

- אם $x \in \bar{L}$ אז $x \notin L$ ובמקרה כזה M דוחה את x והמכונה $\overline{M}$ מקבלת את x.
- אם $x \notin \bar{L}$ אז $x \in L$ ובמקרה כזה M מקבלת את x והמכונה $\overline{M}$ דוחה את x.

קיבלנו ש: $\bar{L} = L(\overline{M})$. בנוסף, $\overline{M}$ תמיד עוצרת (כי M עוצרת תמיד).

מכאן, שקיימת מכונה מכריעה ל-$\bar{L}$. ולכן $\bar{L} \in R$.

∎

תכונה 3 - המחלקה R סגורה לאיחוד.

טענה 3 - תהיינה $L_1, L_2 \in R$ אז $L_1 \cup L_2 \in R$.

הוכחה 3 - תהיינה L_1, L_2 שייכות ל-R, ותהיינה M_1, M_2 מ"ט המכריעות את L_1, L_2 בהתאמה.

נגדיר מכונה חדשה M' על קלט x שמבצעת:

1. M' מריצה את M_1 על x ואם M_1 קיבלה, אזי M' עוצרת ומקבלת.
2. אחרת, M' מריצה את M_2 על x ועונה כמוה.

נראה נכונות עבור שפת האיחוד:

- אם $x \notin L_1 \cup L_2$ אז $x \notin L_2 \wedge x \notin L_1$ כלומר $M_2(x) = 0 \wedge M_1(x) = 0$ ולכן $M'(x) = 0$.
- אם $x \in L_1 \cup L_2$ אז $x \in L_2 \vee x \in L_1$ כלומר $M_2(x) = 1 \vee M_1(x) = 1$ ולכן $M'(x) = 1$.

כלומר הראנו ש: $L(M') = L_1 \cup L_2$. הראנו שקיים מ"ט M' מכריעה לשפת האיחוד ומכאן $L_1 \cup L_2 \in R$.

∎

תכונה 4 - המחלקה RE סגורה לאיחוד.

טענה 4 - תהיינה $L_1, L_2 \in RE$ אז $L_1 \cup L_2 \in RE$.

מבוא להוכחה 4 - אנחנו לא יכולים להוכיח כמו שהוכחנו עבור R מכיוון שב-RE קיימים מקרים של קלטים לא מוגדרים על המכונות. אנחנו נבצע הרצה מבוקרת כמו שראינו שניתן לסמלץ מכונה בהינתן קידוד עבורה.

ניתן גם לסמלץ שתי מכונות במקביל. לדוגמה - בכל צעד זוגי, להריץ את המכונה הראשונה ובכל צעד אי-זוגי להריץ את המכונה השנייה. נוכל להשתמש ב-2 סרטים - אחד עבור ריצת המכונה הראשונה ואחד עבור ריצת המכונה השנייה.

במילים אחרות, כאשר המכונה האוניברסלית U מקבלת קלט הוא יושב אצלה בסרט אחד.
אבל המכונה U יכולה:

1. להשאיר את החצי הראשון של הקלט בסרט 1 עם הקונפיגורציה שלה בסרט 2.
2. לגזור את החצי השני של הקלט לסרט 3 ואת הקונפיגורציה שלה בסרט 4.

<u>רעיון הוכחה 4</u> - תהיינה $L_1,\ L_2$ שייכות ל-RE, ותהיינה $M_1,\ M_2$ מ"ט המקבלות את L_1, L_2 בהתאמה.
נגדיר מכונה חדשה M' על קלט x שמבצעת:

1. $M'(x)$ תסמלץ את שתי המכונות $M_1,\ M_2$ <u>במקביל</u> על x.
2. אם אחת מהן עצרה ב-q_{accept} אז מיד נעצור ו<u>נקבל</u>.
3. אם שתיהן עצרו ב-q_{reject} אז נעצור ו<u>נדחה</u>.
4. אחרת - המכונה שלנו פשוט תמשיך לרוץ לנצח (וזה בסדר...).

נשאר להוכיח את נכונות הבניה. כלומר ש: $x \in L(M_1)$ או $x \in L(M_2)$.

- $x \in L_1 \cup L_2$ אז לפחות אחת מהמכונות <u>עוצרת</u> על x ב-q_{accept} ונסמן ב-i את הצעד בו היא עצרה. מהבניה של M', המכונה M' תזהה עצירה זו בסימולציה של הצעד הנ"ל בצעד סימולציה $2i$ לכל היותר (כי יש 2 סרטים שרצים במקביל). במקרה זה, היא גם תעצור ותקבל. גם אם M_2 עצרה לפני כן, זה לא ישנה את התוצאה ל-q_{reject} כי M' לא עוצרת ב-q_{reject} אלא אם שתי המכונות עצרו ב-q_{reject}.
- $x \notin L_1 \cup L_2$ אז $x \notin L_1 = L(M_1)$ וגם $x \notin L_2 = L(M_2)$. כלומר, אף אחת מהמכונות לא תעצור במצב q_{accept}. לכן M' גם לא תעצור ב-q_{accept} אלא תעצור ב-q_{reject} אם שתיהן עוצרות או לא.

∎

<u>תכונה 5</u> - המחלקה RE סגורה לחיתוך.
<u>טענה 5</u> - תהיינה $L_1,\ L_2 \in RE$ אז $L_1 \cap L_2 \in RE$.
בהינתן $L_1, L_2 \in RE$ המתקבלות על ידי מ"ט M_1, M_2, נקבל את $L_1 \cap L_2$.
נבנה מ"ט דו-סרטית M אשר בהינתן קלט x תבצע:

1. M תעתיק את x לסרט נוסף.
2. תריץ את M_1 על x בסרט הראשון.
   a. אם M_1 דחתה, M <u>תדחה</u>.
   b. אם M_1 קיבלה אז M תריץ את M_2 על x בסרט השני ותענה כמוה.

אזי M תקבל קלט x אם"ם x מתקבל ע"י M_1 וגם ע"י M_2. כלומר: $L(M) = L_1 \cap L_2$.
נותר להוכיח את נכונות הבניה.

∎

סגירויות של המחלקות $R,\ RE$

כל אחת מהטענות הללו דורשות הוכחה. (לחלקן צריך הרצה מבוקרת).

סגירות	מחלקה R	מחלקה RE
משלים	✔	✘
איחוד	✔	✔
חיתוך	✔	✔
שרשור	✔	✔
איטרציה	✔	✔
היפוך (רוורס)	✔	✔

המחלקה co-RE

<u>הגדרה</u>: $coRE = \left\{L \mid \bar{L} \in RE\right\}$. כלומר, קבוצת כל השפות אשר לשפה המשלימה שלהן יש מכונה <u>מקבלת</u>.

<u>הגדרה שקולה</u>: הקבוצה $coRE$ היא קבוצת השפות שיש עבורן מכונה M כך ש:

1. אם $x \in L$ אז המכונה עוצרת ומקבלת או לא עוצרת בכלל.
2. אם $x \notin L$ אז המכונה עוצרת ודוחה.

<u>אבחנה</u>: כשמגדירים שפה $L \in coRE$ על ידי מכונה M כנ"ל, לא בהכרח מתקיים $L(M) = L$.
שאלה - מתי זה כן מתקיים. תשובה - עבור כל שפה $L \in R \subseteq coRE$.

<u>סדר בראש</u>:

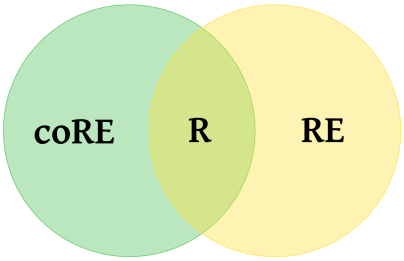
עבור $L \in RE$ מתקיים:

- אם $x \in L$ אז $M(x) = 1$ <u>ועוצר</u>.
- אם $x \notin L$ אז $M(x) = 0$ או $M(x)$ לא עוצר.

עבור $L \in coRE$ מתקיים:

- אם $x \in L$ אז $M(x) = 1$ או $M(x)$ לא עוצר.
- אם $x \notin L$ אז $M(x) = 0$ <u>ועוצר</u>.

סיכום מחלקות עד כה



$R = \{L \mid L$ את המכריעה M מכונה מ"ט קיימת$\}$

$RE = \{L \mid L$ את המקבלת M מכונה מ"ט קיימת$\}$

$coRE = \{L \mid L$ את הדוחה M מכונה מ"ט קיימת$\}$

תכונות של המחלקה coRE

<u>תכונה</u>: $R \subseteq coRE$: מ"ט המכריעה שפה L, בהכרח עוצרת על כל קלט $x \notin L$.

<u>טענה</u>: $R = RE \cap coRE$.

<u>הוכחה</u>: נוכיח בהכלה דו-כיוונית.

- <u>כיוון ראשון</u> $R \subseteq RE \cap coRE$:

ראינו ש: $R \subseteq RE$ וגם ש: $R \subseteq coRE$ מכאן $R \subseteq RE \cap coRE$.

- <u>כיוון שני</u>: $RE \cap coRE \subseteq R$:

תהי $L \in RE \cap coRE$, צריך להוכיח כי $L \in R$ כלומר שיש לשפה L מכונה מכריעה.

מכיוון ש: $L \in RE$ אז קיימת לה מכונה M_1 המקבלת את L.

וגם $L \in coRE$ אז קיימת לה מכונה M_2 שמזהה קלטים שלא ב-L ודוחה אותם.

(זאת בעצם המכונה **המקבלת** של השפה $\bar{L}$).

נתאר מכונה חדשה M^* בהרצה מבוקרת. המכונה M^* על קלט x:

1. מריצה **במקביל** את M_1 על x ואת M_2 על x.
2. בכל שלב, אם M_1 קיבלה אז M^* גם מקבלת. אם M_2 דחתה אז M^* גם דוחה.

נשאר להוכיח את נכונות הבניה:

- אם $x \in L$ אז $M_1(x) = 1$ כלומר $M^*(x) = 1$.
- אם $x \notin L$ אז $M_2(x) = 0$ כלומר $M^*(x) = 0$.

<u>הערת עזר לכיוון שני של ההוכחה הקודמת</u>:

$L \in RE$ אז קיים לה מ"ט M_1 שעוצרת במצב קבלה **וגם** $L \in coRE$ אז קיים לה מ"ט M_2 שעוצרת במצב דחייה. אמרנו שכל $L \in R$ קיים לה מ"ט M שעוצר תמיד וקובע אם קיבל או דחה. לכן, אנחנו צריכים "לתפוס" את המקרים שבהם M_1, M_2 עוצרים. לכן, ניצור M^* ונראה בהרצה מבוקרת שכל מילה מתקבלת $L \in x$ אז נפנה ל-M_1 שעוצר במצב מקבל. ואם $x \notin L$ אז נפנה ל-M_2 שעוצר במצב דוחה. הראנו בנייה עבור מ"ט M^* <u>שמכריעה</u> את השפה ולכן $L \in R$.

השפה L_u

האם השפה $L_u = \{< M, x > \mid M\ accepts\ x\}$ שייכת ל-RE? האם היא שייכת ל-R? - הוכיחו.

<u>טענה</u>: $L_u \in RE$.

<u>הוכחה</u>: נתאר מכונת טיורינג M_u המקבלת את השפה L_u:

M_u על קלט $< M, x >$ תסמלץ את ריצת M על הקלט x ותענה כמוה.

- (ראינו שניתן "לסמלץ" ריצה של מכונה שנתונה לנו כקידוד).

נשאר להוכיח את נכונות הבניה:

- אם $< M, x > \in L_u$ אז M מקבלת את הקלט x, מכאן ש-M_u מקבלת את הקלט $< M, x >$ אזי $< M, x > \in L(M_u)$.
- אם $< M, x > \notin L_u$ אז M לא מקבלת את הקלט x, מכאן ש-M_u לא מקבלת את הקלט $< M, x >$ אזי $< M, x > \notin L(M_u)$. (כלומר היא דוחה או לא עוצרת כהגדרה של RE).

הראנו ש: $L_u = L(M_u)$ כלומר שקיימת מכונה M_u <u>המקבלת</u> את השפה L_u ולכן $L_u \in RE$.

∎

<u>האם השפה L_u שייכת למחלקה R?</u>: התשובה היא לא כלומר $L_u \notin R$.

כדי להוכיח זאת אנחנו צריכים להוכיח **שכל** אלגוריתם בעולם, **לא** יכול להכריע את השפה הזו. (נראה בהמשך).

השפה L_D

האם השפה $L_D = \{< M > \mid M\ accepts\ < M >\}$ שייכת ל-RE? האם שייכת ל-R? - הוכיחו.

<u>טענה</u>: $L_D \in RE$.

<u>הוכחה</u>: נתאר מ"ט M_D המקבלת את השפה L_D:

M_D על הקלט $< M >$ תסמלץ את ריצת M על הקלט $< M >$ ותענה כמוה.

נשאר להוכיח את נכונות הבניה:

- אם $< M > \in L_D$ אז $M_D(< M >) = 1$.
- אם $< M > \notin L_D$ אז $M_D(< M >) = 0$ או $M_D(< M >)$ לא תעצור (לא מוגדרת).

הראנו ש: $L_D = L(M_D)$ כלומר שקיימת מכונה M_D <u>המקבלת</u> את השפה L_D ולכן $L_D \in RE$.

שיטת הלכסון של קנטור

קנטור הוכיח שבקטע $[0, 1]$ יש כמות **לא** בת מניה של מספרים.

<u>מבנה ההוכחה</u>:

1. נניח בשלילה שיש מספר בן-מניה של מספרים בין 0 ל-1.
2. לשם הנוחות, ניקח תת-קבוצה A של מספרים בין 0 ל-1, שהייצוג שלהם מכיל רק 2 תווים (למשל 0,1).
3. נסדר את המספרים הללו בסידור כלשהו (מובטח שקיים סידור כזה מכיוון שהנחנו שמדובר בקבוצת בת מנייה).
4. נבנה מספר חדש c על ידי זה שניקח את הספרות שבאלכסון ונחליף אותן, 1 יהפוך ל-0 ו-0 יהפוך ל-1.
5. המספר $c \in A$ ולכן קיים אינדקס i כך שהמספר שלנו הוא בדיוק האיבר n_i.
6. אבל, לפי הבנייה של c, הערך של c בתו ה-i שונה מהערך של n_i בתו ה-i.
7. סתירה.

<u>הרעיון מאחורי ההוכחה</u>:

יש לנו אינסוף מספרים בין 0 ל-1, אבל כמות בת מניה של מספרים ולכן אנחנו יכולים לתת למספרים האלה סידור, כל מספר קיבל ייצוג n_j בהתאם לסידור הזה. לכן, בין האינסוף מספרים האלה קיים ייצוג n_i של המספר החדש c שנוצר מההחלפה שעשינו. אבל בייצוג הזה קיבלנו ש: $n_i[i] \neq c[i]$. כלומר n_i לא מייצג את c.

אבל מכיוון שקיים סידור (בן מניה) והמספר לא נמצא בשורה הזאת אז הוא לא קיים בין 0 ל-1 ו-A לא בן מניה.

האם השפה L_D שייכת ל-R?

האם השפה $L_D = \{< M > \mid M\ accepts\ < M >\}$ שייכת ל-R?

<u>טענה</u>: $L_D \notin R$.

<u>מבנה ההוכחה</u>: ניצור <u>פרדוקס</u>: נניח בשלילה ונראה שזה גורר סתירה.

<u>הוכחה</u>: נניח בשלילה ש: $L_D \in R$ ולכן קיימת לשפה L_D מכונה M_D <u>המכריעה</u> אותה.

נגדיר מכונה חדשה A_D (היא גם מכונה מכריעה לפי סגירות למשלים) שפועלת בצורה הבאה:

המכונה A_D על הקלט $< M >$ מריצה את המכונה M_D על הקלט $< M >$ ועונה <u>הפוך</u>. (כמו שהפכנו את הביטים בלכסון של קנטור). מכיוון שיש מספר בן מניה של מכונות טיורינג, ניתן לסדר את כל המכונות בסידור כלשהו (למשל סידור לקסיקוגרפי של מחרוזות הקידוד).

נבנה טבלה אינסופית של כל הזוגות האפשריים של מכונות טיורינג, כך שבתא ה-$[i][j]$ נאחסן את התשובה לשאלה הבאה: האם המכונה M_i מקבלת את הקלט $< M_j >$ (הקידוד של המכונה M_j)?

	$< M_1 >$	$< M_2 >$	$< M_3 >$	$< M_4 >$	...	...	...
$< M_1 >$	0	0	0	0	1	1	
$< M_2 >$	1	1	1	0	0	0	
$< M_3 >$			1				
$< M_4 >$				0			
...					0		
...						1	
...							0

המכונה M_D יודעת למלא את התאים של האלכסון בטבלה (כי M_D מכריעה את השפה L_D שמקבלת כל קידוד $< M_i >$ של המכונה M_i כלומר כל $[i][i]$ בונה את האלכסון של הטבלה), לכן המכונה A_D גם כן יודעת למלא את תאי האלכסון של הטבלה (ממלאת את האלכסון אבל בדיוק ההפך).

המכונה A_D היא גם מכונת טיורינג ולכן היא נמצאת בטבלה, כלומר קיים אינדקס i כך שמתקיים ש: $A_D = M_i$.

מה הערך בתא ה-$[i][i]$ בטבלה? - כלומר מה הערך של: $A_D(< M_i >) = M_i(< M_i >)$?

- אם A_D מקבלת על הקלט $< A_D >$ אז בתא $[i][i]$ צריך להיות ערך 1 (זה נכון לפי הגדרת הטבלה).
  אבל בהרצה של A_D על הקלט $< A_D >$, המכונה M_D תענה 1 ואז A_D חייב לפלוט 0. - סתירה.
- אם A_D לא מקבלת על הקלט $< A_D >$ אז בתא $[i][i]$ צריך להיות ערך 0 (זה נכון לפי הגדרת הטבלה).
  אבל בהרצה של A_D על הקלט $< A_D >$, המכונה M_D תענה 0 ואז A_D חייב לפלוט 1. - סתירה.

∎

השפה HP (Halting Problem)

האם $HP = \{< M, x > \mid M \text{ halts (stops) on } x\}$ שייכת ל-RE? האם שייכת ל-R? - הוכיחו.

אינטואיציה: אם M עצרה על x אז $< M, x > \in HP$, כלומר הגענו למצב מקבל q_{accept} והקלט בשפה.
אם M לא עצרה על x אז $< M, x > \notin HP$, אבל אנחנו לא יודעים אם בוודאות המכונה עצרה ולכן, המכונה M לא יכולה להכריע את השפה. מכאן - $HP \in RE$ אבל $HP \notin R$.

טענה: $HP \in RE$.

רעיון ההוכחה: נראה מכונה שאם היא עוצרת אז היא מקבלת את הקלט - אחרת היא לאו דווקא עוצרת.

הוכחה: נתאר מכונת טיורינג M_{HP} המקבלת את השפה HP:

המכונה M_{HP} על הקלט $< M, x >$ תסמלץ את ריצת M על הקלט x, אם עצרה - נקבל (נעבור ל-q_{accept}).
אם הקידוד $< M, x >$ לא חוקי - המכונה M_{HP} לא עוצרת.

נשאר להוכיח את נכונות הבניה:

- אם $< M, x > \in HP$ אז M עוצרת על הקלט x ולכן M_{HP} מקבלת את הקלט $< M, x >$.
  אזי $< M, x > \in L(M_{HP})$.
- אם $< M, x > \notin HP$ אז M לא עוצרת על הקלט x ולכן M_{HP} לא מקבלת את הקלט $< M, x >$.
  ובפרט, המכונה M_{HP} לא עוצרת על קלט זה. אזי $< M, x > \notin L(M_{HP})$.

הראנו ש: $HP = L(M_{HP})$ כלומר שקיימת מכונה M_{HP} המקבלת את השפה HP ולכן $HP \in RE$.

∎

בעיית העצירה

שאלה: האם השפה $HP \in R$?

תשובה: לא. כלומר $HP \notin R$.

רעיון ההוכחה:

1. נניח בשלילה שיש מכונה M_{HP} המכריעה את HP.
2. נשתמש ב-M_{HP} כדי להכריע את $L_D = \{< M > \mid M\ accepts\ < M >\}$.
3. אבל הוכחנו שלא קיימת מכונה M_D שמכריע את השפה L_D.
4. לכן גם לא קיימת מכונה שמכריע את HP - נקבל סתירה.

טענה: $HP \notin R$.

הוכחה: נניח בשלילה $HP = \{< M, x > \mid M\ halts\ (stops)\ on\ x\} \in R$ ולכן קיימת מכונת טיורינג M_{HP} המכריעה את השפה HP. בהינתן קידוד של מכונה כלשהי $< M >$, נרצה **להכריע** האם היא שייכת לשפה $L_D = \{< M > \mid M\ accepts\ < M >\}$ או לא:

נבנה מכונה M' אשר זהה ל-M, רק שבכל פעם כאשר M דוחה אז M' נכנסת ללולאה אינסופית.
עבור המכונה M' על הקלט y :

- מסמלצת את ריצת M על y, אם M קיבלה - אז M' מקבלת.
- אם M דחתה - אז M' נכנסת ללולאה אינסופית.

כעת נשלח אל המכונה M_{HP} את הקידוד $< M', < M > >$.

- אם M_{HP} תענה שהקלט בשפה HP אז M' עוצרת על $< M >$ ולפי בניית M' נסיק ש-M מקבלת את $< M >$ ונסיק מכך כי $< M > \in L_D$.
- אם M_{HP} תענה שהקלט לא בשפה HP אז M' לא עוצרת על $< M >$ ולפי בניית M' נסיק ש-M *לא-מקבלת* את $< M >$ (דוחה או נכנסת ללולאה אינסופית) ונסיק מכך כי $< M > \notin L_D$.

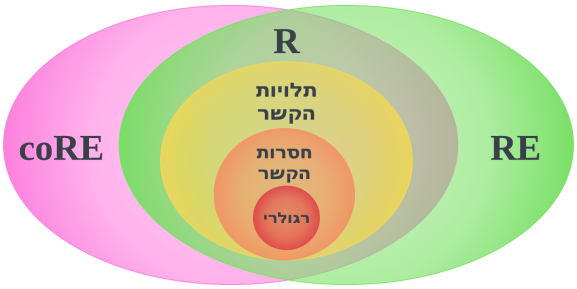
קיבלנו אלגוריתם **שמכריע** את השפה L_D - סתירה.

∎

<u>הערות הוכחה</u>:

- אם יש לנו מכונה M_{HP} שמכריע את HP אז המכונה הזאת אומרת לנו האם נכנסו ללולאה אינסופית או לא (האם עצרנו או לא). אם יש לולאה אינסופית אז נדחה ואם הגענו למצב סופי אז נקבל.
- הסתירה נובעת כי הוכחנו כבר ש: $L_D \notin R$ כלומר שאי אפשר להכריע אותה אבל הראנו הרגע שקיים אלגוריתם שמכריע אותה וזו הסתירה.
- הוכחנו ישירות משיטת הלכסון של קנטור ש: $L_D \notin R$ ואנחנו נעזרים בו בשביל להוכיח ש: $HP \notin R$. כלומר הוכחנו ש: HP לא כריעה.

תמונת מצב



בקורס אוטומטים למדנו על המחלקות הרגולריות, חסרות ההקשר ותלויות ההקשר. לכל אחת מהמחלקות האלה קיים מכונה שיודעת למדל אותם. כל מכונה שונה כזאת חזקה יותר מהשניה ככל שהיא תכיל יותר מחלקות בתוכה.

<u>השפה הרגולרית</u> - הם שפות שלא צריך לזכור בהן משהו אינסופי. הן שפות מחזוריות. כלומר: לא צריך ספירה מדויקת וגם לא צריך למנות משהו כנגד משהו אחר. לדוגמה השפה $L = \{a^i : i \in \mathbb{N}\}$ היא שפה רגולרית. לשפות האלו למדנו את המודלים $DFA,\ NFA,\ NFA - \varepsilon$ וקבענו כי אם ניתן לבנות לשפה אוטומט ממודלים אלה אז השפה היא רגולרית. בתחילת הקורס של אוטומטים, הסבירו לנו כי לכל אוטומט יש סרט וראש קורא שעובר על הקלט. עבור כל תו שהראש קורא בסרט, נשתמש בפונקצית המעברים δ כדי לקבוע (דטרמיניסטי או לא) לאיזה מצב להמשיך. כלומר, המודלים $DFA,\ NFA,\ \varepsilon NFA$ הם מקרה פרטי של מכונת טיורינג שיודעות לקבל רק את השפות הרגולריות ולא יותר מזה. לפי הציור אפשר להבין כי כל השפות רגולריות הם גם חסרות הקשר, תלויות הקשר, $R,\ RE$.

<u>שפות חסרות ההקשר</u> - הם שפות שיש להן זיכרון של מחסנית. לא ניתן לבצע ספירה מדויקת אבל ניתן לזכור כמות כנגד כמות. לדוגמה, השפה $L = \{a^n b^n : n \in \mathbb{N}\}$ היא שפה חסרת הקשר ובפרט לא רגולרית. לשפות אלה למדנו את המודל PDA או $Push\ Down\ Automata$ כלומר אוטומט המחסנית. למדנו על דקדוקים חסרי הקשר והראנו שהם שקולים ל-PDA. גם כאן, למדנו שלאוטומט PDA יש ראש קורא/כותב וסרט. כלומר PDA הוא גם מקרה פרטי של מכונת טיורינג שמחקה התנהגות של מחסנית. מודל PDA הוא מודל חזק יותר ממודלי DFA של השפות הרגולריות וככל שנעטוף יותר מחלקות ככה המודל יהיה חזק יותר.

<u>שפות תלויות הקשר</u> - הם שפות שאפשר לבצע בהן ספירה מדויקת ותלות משולשת בין תווים. לדוגמה השפות $L = \{a^n b^n c^n : n \in N\}$ ו- $L = \{a^p : p\ is\ prime\}$ הם תלויות הקשר. לשפות תלויות הקשר יש להן זיכרון חסום של מכונת טיורינג (מקרה פרטי) והם חזקות יותר ממודלי $DFA,\ PDA$ וכל מחלקה שהיא עוטפת.

<u>המחלקה R</u> - היא קבוצת כל השפות שקיים עבורן מכונת טיורינג שמכריעה אותן. כלומר, זה שפות שאפשר לקבוע עבורן החלטה חד-משמעית האם המילה מתקבלת או לא. למשל השפה Σ^* נמצאת ב-R מכיוון שלא ניתן לקבוע חישובים מסובכים, עבור כל $x \in \Sigma^*$ אז $M(x) = 1$ אם תתקבל או $M(x) = 0$ אם תדחה. מכיוון שהמכונה M מכירה את Σ אז אין לנו כאן בעיות שעלולות להוביל אותנו לריצה אינסופית.

<u>המחלקה RE</u> - היא קבוצת כל השפות שקיים עבורן מכונת טיורינג שמקבלת אותן. זה שפות שייתכן שקיים להן מכונת טיורינג שעבור קלט מסוים עלולות להוביל את המכונה למצב אינסופי ולא להיעצר לעולם.

סיכום הרצאה 2

<u>מכונת טיורינג אוניברסלית</u> - בתחילת ההרצאה למדנו על מכונת טיורינג אוניברסלית U שהיא מכונת טיורינג כללית שתוכל לבצע את המשימה של כל מכונת טיורינג אחרת. היא צריכה לקבל קידוד של מכונה M ואת הקלט $x \in \Sigma^*$ עבור M ותחזיר $M(x)$. מכונת טיורינג אוניברסלית U מחקה את ריצת M על x בעזרת שמירת קונפיגורציה בסרט נוסף, כאשר הקונפיגורציה נקבעת על סמך פונקציית המעברים δ שנתונה בקידוד $< M >$. כאשר M תגיע למצב סופי ותעצור אז U תחזיר $M(x)$.

<u>מכונת טיורינג לקבלת שפות</u> הינה מכונת טיורינג רגילה M שמקיימת $F = \{q_{acc}, q_{rej}\}$.

- נאמר ש: M <u>מקבלת</u> את x ונסמן $M(x) = 1$.
- נאמר ש: M <u>דוחה</u> את x ונסמן $M(x) = 0$.
- שפת המכונה $L(M)$ הינה קבוצת כל המילים אשר המכונה M מקבלת אותם $(M(x) = 1)$.
- השפה המשלימה $\overline{L(M)} = \Sigma^* \setminus L(M)$ היא קבוצת כל המילים ש-M דוחה **או** לא עוצרת.

<u>קבלת שפה</u> - נאמר שמכונה M_L **מקבלת (מזהה)** את השפה L אם מתקיים: $x \in L \leftrightarrow M(x) = 1$.

<u>הכרעת שפה</u> - נאמר שמכונה M_L **מכריעה** את השפה L אם מתקיים: $x \in L \leftrightarrow M(x) = 1$.

בנוסף, M_L **תמיד** עוצרת.

<u>המחלקה R</u> - היא קבוצה שמכילה את כל השפות שקיימות עבורן מכונת טיורינג **המכריעה** אותן.

<u>המחלקה RE</u> - היא קבוצה שמכילה את כל השפות שקיימות עבורן מכונת טיורינג **המקבלת** אותן.

1. $R \subseteq RE$: מ"ט המכריעה שפה, בהכרח גם מקבלת אותה.
2. המחלקה R סגורה למשלים: אם $L \in R$ אז $\bar{L} \in R$.
3. המחלקה R סגורה לאיחוד.
4. המחלקה RE סגורה לאיחוד.
5. $coRE = \{L \mid \bar{L} \in RE\}$. כלומר, קבוצת כל השפות אשר לשפה המשלימה שלהן יש מכונה <u>מקבלת</u>.
6. $R \subseteq coRE$: מ"ט המכריעה שפה L, בהכרח עוצרת על כל קלט $x \notin L$.
7. $L_u = \{< M, x > \mid M\ accepts\ x\} \in RE$.
8. $L_D = \{< M > \mid M\ accepts\ < M >\} \in RE$
9. $HP = \{< M, x > \mid M\ halts\ (stops)\ on\ x\} \in RE$

<u>שיטת הלכסון של קנטור</u> - קנטור הוכיח שבקטע $[0, 1]$ יש כמות **לא** בת מניה של מספרים. נשתמש בשיטה זו כדי להראות אי שייכות למחלקה R.

הרצה מבוקרת

נציג כעת שיטה חזקה להוכחה ששפות מסוימות שייכות ל-RE.

בתור דוגמה, נתונה השפה הבאה: $L = \{< M > \mid \exists x\ s.t.\ f_M(x) = x\}$.

- זוהי שפת כל המכונות שיש "נקודת שבת" לפונקציה שהן מחשבות.
- בהינתן דקדוק מכונה $< M >$, לא ברורה הדרך לבדוק האם קיימת נקודת שבת מלבד לבצע $Brute\ Force$ לכל קלט $w_i \in \Sigma^*$ אפשרי כזה (יש אינסוף כאלה).
- במכונות המקבלות שפה, ייתכן שעבור קלט מסוים המכונה לא תעצור לעולם.
- מה הבעיה? - יכול להיות ש: $M(w_2) = w_2$ אבל $M(w_1)$ לא תעצור לעולם ולכן M לא תבדוק את w_2.
- פתרון - נשתמש ב**הרצה מבוקרת**.

<u>מה זה הרצה מבוקרת?</u> - כדי להימנע ממצב שעבור קלט מסוים w, המכונה תיכנס ללולאה אינסופית, נשתמש בשיטת הרצה מבוקרת. זו שיטה עדינה ומבוקרת יותר המונעת ריצה אינסופית. בכל צעד באלגוריתם, נפעיל חלק קטן מתוך כל מילה w_i שנעבוד עליה. ככל שמספר הצעדים גדל - ככה נעבוד על יותר מילים במקביל - כך שאם מילה w_i תיכנס למצב אינסופי, המילה w_{i+1} שגם נעבוד עליה במקביל, תמשיך להיבדק עד לעצירתה. נסתכל על הפתרון הבא שיעזור לנו להבין יותר את רעיון השיטה הזו.

<u>שאלה</u>: האם $L = \{< M > \mid \exists x\ s.t.\ f_M(x) = x\} \in RE$? - הוכיחו.

<u>פתרון</u>: כן, $L \in RE$. נראה בהרצה מבוקרת.

נבנה מכונה M_L המקבלת את השפה L.

M_L על קלט $< M >$:

- נסדר את כל המילים ב-Σ^* בסדר לקסיקוגרפי $w_1, w_2, w_3, w_4, \ldots$. (אפשרי כי בת מניה).
- עבור $i = 1$; עד אינסוף ; i ++:
  - עבור $j = 1$; עד i ; j ++:
    - סמלץ את M על קלט w_j במשך i צעדים. (כלומר $M\left(w_j[1\ldots i]\right)$).
    - אם M החזירה w_j, <u>קבל</u>.

נותר להוכיח את נכונות הבניה:

<u>כיוון ראשון</u> - יהי $L \ni < M >$, אז קיים x כך ש: $f_M(x) = x$.

- נסמן את המילה ה-k כך ש: $x = w_k$ (כלומר x זו המילה ה-k בסדר לקסיקוגרפי).
- נסמן t מספר הצעדים של M על x (כלומר אחרי t צעדים, M <u>עוצרת</u> ומחזירה x).
- נסמן $i = max(t, k)$ (אנחנו נרצה $i \geq k$ כדי שנריץ את המילה w_k וגם $i \geq t$ צעדים עד ש-M תסיים לעבור על w_k).
- נרוץ על $w_1, \ldots, w_i$ (ובטוח גם על w_k) למשך i צעדים (לפחות t צעדים) ולכן נזהה את $f_M(x) = x$ ונקבל את $< M >$.

- לכן, $M_L(< M >) = 1$, כלומר $< M > \in L(M_L)$.

כיוון שני - יהי $< M > \notin L$, לא קיים x כך ש: $f_M(x) = x$.

- לא קיים k כך ש: $f_M(w_k) = w_k$.
- לכל i ולכל j, המכונה M לא תעצור ותחזיר את w_j.
- המכונה M_L תרוץ לנצח (כי היא לא תמצא אף מילה שמקיימת את f_M).
- לכן, $< M > \notin L(M_L)$.

הסבר כיוון ראשון: נניח שהמילה ה-$k = 1$, כלומר ה-w_1, עוצרת (מסתיימת) בריצת המכונה M לאחר $t = 30$ צעדים והיא אכן מקיימת ש- $f_M(w_1) = w_1$ (בהכרח יש כזאת כי $< M > \in L$).
כלומר, צריך לקרוא את כל המילה w_1 בשביל לקבוע אם מתקיים. המילה הזאת היא בת כ-$t = 30$ תווים.
רק כאשר באלגוריתם שלנו, ה-i (לולאה החיצונית) יהיה $i = max\{t, k\}$ אז M תסיים לקרוא את המילה w_1 ותקבע ש $f_M(w_1) = w_1$. כלומר אנחנו נצטרך בזמן הריצה לעבור על $w_1,...,w_i$ מילים עד ש-w_1 תעצור.

תרגיל

נתונה השפה הבאה: $L = \{< M > \mid < M > \notin L(M)\}$. האם היא ב-$RE$? האם ב-$R$? האם ב-$coRE$?

האם היא ב-R? - לא. אם היא הייתה, אז ניתן היה בקלות להכריע את L_D שראיתם בהרצאה שהיא לא ב-R.

האם היא ב-coRE? - כן. נוכיח את זה. צריך להראות מכונה כך ש:

1. אם הקלט לא בשפה, אז המכונה תמיד עוצרת ודוחה.
2. אם הקלט בשפה, אז המכונה מקבלת או לא עוצרת.

נבנה מכונה M' המקבלת את השפה L.
מכונה M' על קלט $< M >$ תבצע:

1. נסמלץ את M על $< M >$.
2. אם M עצרה:
   a. וקיבלה את הקלט, אז M' תדחה.
   b. ודחתה את הקלט, אז M' תקבל.

נוכיח את נכונות הבניה.
כיוון ראשון - $< M > \in L$, נראה כי $< M > \in L(M')$.

1. לפני ההנחה M עוצרת ודוחה או לא עוצרת על $< M >$.
2. לכן, M' תעצור ותקבל או לא תעצור. (כי אם M לא עצרה אז גם M' לא עוצרת).
3. מכאן $< M > \in L(M')$.

כיוון שני - $< M > \notin L$, נראה כי $< M > \notin L(M')$.

1. לפי ההנחה, M עוצרת ומקבלת את $< M >$.
2. לכן, M' תעצור ותדחה.
3. כלומר, $< M > \notin L(M')$ כנדרש.

∎

האם היא ב-RE? - לא. הוכחנו כבר ש: $R = coRE \cap RE$. אם $L \in RE$ וגם $L \in coRE$ אז $L \in R$ אבל ראינו כבר ש:$L \notin R$ ולכן $L \notin RE$.

המחלקה R סגורה להיפוך

הגדרה: בהינתן שפה L, נגדיר את השפה $L^R = \{w = a_1 a_2 \dots a_n \mid a_n \dots a_2 a_1 \in L\}$.

טענה: תהיה $L \in R$ אז $L^R \in R$. כלומר R סגורה להיפוך.

הוכחה: תהיה שפה $L \in R$ אז קיים לה מכונה M המכריעה אותה.

נבנה מכונה M^R שמכריעה את השפה L^R.

המכונה M^R על קלט $x \in L^R$ תפעל בצורה הבאה:

1. תהפוך את x ל-x^R.
2. תפעיל את M על x^R ותענה כמו M.

נותר להוכיח את נכונות הבניה:

כיוון ראשון - תהיה $x \in L^R$ נוכיח ש: $x \in L(M^R)$.

- לפי הגדרת ההיפוך $x^R \in L$.
- ידוע ש-M מכריעה את L ולכן בהפעלתה על x^R, היא תעצור ותקבל אותה.
- לפי הבניה, M^R עונה אותו הדבר כמו M , אז גם היא תעצור ותקבל.
- קיבלנו ש: $x \in L(M^R)$ כנדרש.

כיוון שני - תהיה $x \notin L^R$, נוכיח כי $x \notin L(M^R)$.

- לפי הגדרת ההיפוך $x^R \notin L$.
- ידוע ש-M מכריעה את L ולכן בהפעלתה על x^R, היא תעצור ותדחה אותה.
- לפי הבניה, M^R עונה אותו הדבר כמו M , אז גם היא תעצור ותדחה.
- קיבלנו ש: $x \notin L(M^R)$ כנדרש.

∎

המחלקה R סגורה לשרשור

<u>הגדרה</u>: בהינתן שתי שפות L_1 ו-L_2, נגדיר את השפה $L_1 \cdot L_2 = \{v \cdot w \mid v \in L_1 , w \in L_2\}$.

<u>טענה</u>: יהיו $L_1, L_2 \in R$ אז $L_1 \cdot L_2 \in R$. כלומר R סגורה לשרשור.

<u>הוכחה</u>: יהיו שתי שפות $L_1, L_2 \in R$. יהיו M_1, M_2 מ"ט המכריעות את L_1, L_2 בהתאמה.

נבנה מכונה חדשה M' שתכריע את השפה $L_1 \cdot L_2$.

M' על הקלט $x \in L_1 \cdot L_2$ תבצע:

1. הקלט x מורכב מהתווים: $x = a_1 a_2 a_3 .... a_n$.
2. לכל $i = 1, 2, ..., n$ בצע:
   a. נציג פירוק למילה: $x = v \cdot w$ כאשר: $w = a_{i+1} .... a_n \quad , v = a_1 .... a_i$.
   b. נריץ את M_1 על v.
   c. נריץ את M_2 על w.
   d. אם $M_1(v) = 1$ וגם $M_2(w) = 1$ (שתיהן קיבלו) אז $M'(x) = 1$, כלומר <u>נקבל</u>.
3. אם לא קיבלנו לכל $1 \leq i \leq n$ אז $M'(x) = 0$, כלומר <u>נדחה</u>.

נותר להוכיח את נכונות הבניה.

<u>כיוון ראשון</u> - תהיה $x \in L_1 \cdot L_2$. נוכיח כי $x \in L(M')$.

- לפי ההנחה, קיימות מילים $w \in L_2$ ו-$v \in L_1$ כך ש: $x = v \cdot w$.
- בשלב מסוים בריצה, M' תבחן את החלוקה הנכונה עבור x:
  - M_1 <u>תקבל</u> את v כי היא מכריעה את L_1.
  - M_2 <u>תקבל</u> את w כי היא מכריעה את L_2.
- לכן, לפי הבנייה של M' היא <u>תקבל</u> כי שתי המכונות גם קיבלו.
- מכאן $x \in L(M')$ כנדרש.

<u>כיוון שני</u> - $x \notin L_1 \cdot L_2$ נוכיח כי $x \notin L(M')$.

- לפי ההנחה, לא קיימות מילים $w \in L_2$ ו-$v \in L_1$ כך ש: $x = v \cdot w$.
- לכן, לכל חלוקה ש M' תבחן בריצה:
  - M_1 <u>תדחה</u> את v כי היא מכריעה את L_1.
  - M_2 <u>תדחה</u> את w כי היא מכריעה את L_2.
- לכן, לפי הבנייה של M' - היא <u>תדחה</u> כי לכל חלוקה הראנו שגם שתי המכונות דחו.
- מכאן $x \notin L(M')$ כנדרש.

∎

המחלקה RE סגורה לשרשור

<u>טענה</u>: יהיו $L_1, L_2 \in RE$ אז $L_1 \cdot L_2 \in RE$. כלומר RE סגורה לשרשור.

<u>מבוא להוכחה</u>: אם נוכיח כמו שעשינו עבור $L_1 \cdot L_2 \in R$ אז האלגוריתם שלנו לא יעצור לכל מילה בשפה.

למה? - כי כאשר נבחר את הפירוק הנכון $x = v \cdot w$ מובטח לנו ש: M_1 ו-M_2 יעצרו, אבל עבור כל פירוק אחר, המכונות M_1 ו-M_2 <u>יכולות</u> לא לעצור. במקרים כאלה, לא נגיע לפירוק הנכון לעולם כי נתקע בלולאה אינסופית.

הפתרון הוא להשתמש בשיטת <u>הרצה מבוקרת</u>.

<u>הוכחה</u>: יהיו שתי שפות $L_1, L_2 \in RE$. יהיו M_1, M_2 מ"ט המקבלות את L_1, L_2 בהתאמה.

נבנה מכונה חדשה M' שתקבל את השפה $L_1 \cdot L_2$.

M' על הקלט x תבצע:

1. הקלט x מורכב מהתווים: $x = a_1 a_2 a_3 .... a_n$.
2. לכל $i = 1, 2, 3, ... \to \infty$ תבצע:
   a. לכל אפשרות של פירוק המילה x ל-2 מילים: $x = v \cdot w$ תבצע:
      i. הרצת M_1 על v למשך i צעדים.
      ii. הרצת M_2 על w למשך i צעדים.
      iii. אם שתיהן קיבלו - אז <u>קבל</u>.
   3. אם עברנו על כל האפשרויות ולא קיבלנו כלל - <u>דחה</u>.

נוכיח את נכונות הבניה.

<u>כיוון ראשון</u> - $x \in L_1 \cdot L_2$, נוכיח $x \in L(M')$.

- לפי ההנחה, למילה x קיים פירוק מהצורה $x = v \cdot w$ כך ש: $v \in L_1, w \in L_2$.
- נניח ש: $M_1(v) = 1$ לאחר k_1 צעדים וגם $M_2(w) = 1$ לאחר k_2 צעדים.
- באיטרציה ה-$i = max\{k_1, k_2\}$, כאשר נבחן את החלוקה הנכונה נקבל ש:
  - M_1 תקבל את v כי היא מקבלת את L_1.
  - M_2 תקבל את w כי היא מקבלת את L_2.
- לכן, M' <u>תקבל</u> כי שתי המכונות קיבלו.
- מכאן $x \in L(M')$.

<u>כיוון שני</u> - $x \notin L_1 \cdot L_2$, נוכיח כי $x \notin L(M')$.

- לפי ההנחה, לא קיים אף פירוק מהצורה $x = v \cdot w$ כך ש: $v \in L_1, w \in L_2$.
- לכל חלוקה ש: M' תבחן ולכל מספר צעדים שנריץ ("או" בין 2 המקרים הבאים):
  - M_1 <u>תדחה</u> או <u>לא תעצור</u> על v כי הוא לא ב-L_1.
  - M_2 <u>תדחה</u> או <u>לא תעצור</u> על w כי הוא לא ב-L_2.
- בכל איטרציה, M' תרוץ על כל האופציות בלי לקבל בשום שלב ריצה.

- המכונה M' לא תסיים את ריצתה לעולם. (כי הלולאה הראשית רצה לאינסוף ונעצרת רק אם נקבל).
- מכאן $x \notin L(M')$ כנדרש.

■

המחלקה coRE סגורה לאיטרציה

<u>טענה</u>: תהי שפה $L \in coRE$, אזי גם $L^* \in coRE$ (סגירות לאיטרציה/סגור-קליין).

<u>הוכחה</u>: תהי שפה $L \in coRE$ אז קיים לה מכונה M שמקבלת אותה.

נבנה מכונה חדשה M^* <u>שתקבל</u> את השפה L^*.

M^* על הקלט x תבצע הרצה מבוקרת "חזקה":

1. נבחר k כך ש: $1 \leq k \leq |x|$ ונבחר פירוק של x ל-k מילים כך: $x = x_1 \dots x_k$ כאשר $x_i \in \Sigma^*$:
2. לכל $i = 1, 2, 3, \dots$ תבצע:
   a. לכל $1 \leq j \leq i$ בצע:
      i. סמלץ את M על x_j על i צעדים.
      ii. אם M דחה, M^* <u>דוחה</u>.
   b. אם M קיבל את כל ה-x_j-ים, M^* <u>מקבל</u>.

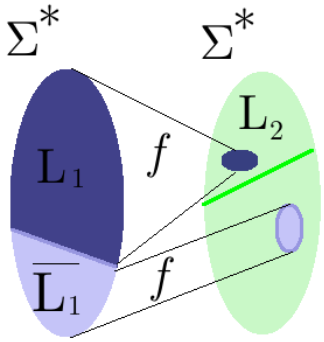
<u>נכונות</u>:

- אם $x \in L^*$ אז קיים פירוק של x לצורה $x = x_1 \dots x_k$ כך שלכל x_j מתקיים $x_j \in L$.
  לכן, יכול להתקיים אחד מהשניים:
  1. המכונה M קיבלה את כל ה-x_j-ים ולכן M^* תקבל.
  2. המכונה M לא עוצרת עבור x_j מסוים וכך גם המכונה M^*.
- אם $x \notin L^*$ אז לכל פירוק של x לצורה $x = x_1 \dots x_k$ קיים x_j כך ש: $x_j \notin L$.
  לכן, $M(x_j)$ ידחה (עבור i גדול מספיק) וגם M^* ידחה באותו האופן.

■

רדוקציות

מבוא



רדוקציה היא שיטה אלגוריתמית המאפשרת להמיר בעיה נתונה לבעיה אחרת שבעזרתה ניתן לפתור את הבעיה המקורית. לדוגמה, נניח שנתונה סדרת מספרים כלשהי, ונניח שקל לפתור את הבעיה "מהו המספר הראשון בסדרה". את הבעיה "מהו המספר הקטן ביותר בסדרה" נוכל לפתור באופן הבא: ראשית נמיין את סדרת המספרים, ואז נפעיל את האלגוריתם שפותר את בעיית "מהו המספר הראשון בסדרה" התוצאה תהיה המספר הקטן ביותר בסדר. משפט רייס מוכיח שהבעיה של זיהוי תכונה לא טריוויאלית של פונקציה היא לא כריעה על ידי רדוקציה מבעיית העצירה (שאינה כריעה) אליה.

<u>בציור</u>: כל הקלטים ב-L_1 ממופים לקלטים של L_2, ואילו קלטים שאינם ב-L_1 ממופים לקלטים שאינם ב-L_2.

תזכורת מהרצאה קודמת

	אם $x \in L$	אם $x \notin L$
$L \in R$ (הכרעה)	מקבל	דוחה
$L \in RE$ (קבלה)	מקבל	דוחה או לא עוצר
$L \in coRE$ (דחייה)	מקבל או לא עוצר	דוחה

בנוסף, למדנו על **בעיית העצירה** עם השפה $HP = \{< M, x > \mid M\ halts\ (stops)\ on\ x\}$.

וראינו כי אם M עצרה על x אז $< M, x > \in HP$, כלומר הגענו למצב מקבל q_{accept} והקלט בשפה.

אם M לא עצרה על x אז $< M, x > \notin HP$, אבל אנחנו לא יודעים אם בוודאות המכונה עצרה.

לכן, המכונה M לא יכולה להכריע את השפה. מכאן - $HP \in RE$ אבל $HP \notin R$.

רדוקציה (הגדרה)

תהיינה שפות $L_1, L_2 \subseteq \Sigma^*$, נאמר ש: L_1 **ניתנת לרדוקציה ל**-L_2 (נסמן $L_1 \leq L_2$)

<u>אם ורק אם</u> קיימת פונקציה $f: \Sigma^* \to \Sigma^*$ שהיא:

1. **<u>פונקציה מלאה</u>**: פונקציה המוגדרת לכל קלט.
2. **<u>פונקציה הניתנת לחישוב</u>**: כלומר, קיימת מכונת טיורינג שיכולה לחשב את הפונקציה הזו.
3. **<u>פונקציה תקפה</u>**: $\forall x \in \Sigma^* : x \in L_1 \Leftrightarrow f(x) \in L_2$.

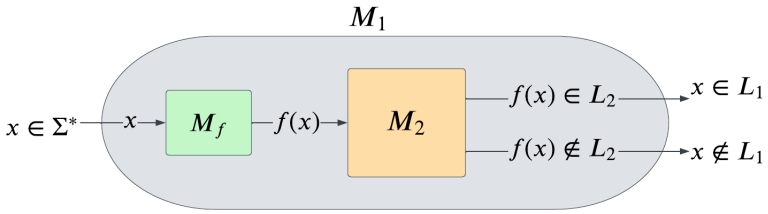
<u>הערות</u>: פונקציה <u>לא</u> מלאה יכולה להיות מהצורה $f(x) = \frac{1}{x}$ כאשר $x = 0$ לא מוגדר.

- פונקציה תקפה - כל קלט x שמתקבל ב-L_1, הפונקציה שלו $f(x)$ צריכה להתקבל ב-L_2.

תקפות של פונקצית רדוקציה

<u>הגדרה</u> - פונקציה תקפה: $\forall x \in \Sigma^* : x \in L_1 \Leftrightarrow f(x) \in L_2$.

<u>באיור</u>: אם נצליח להראות שאם אחרי הפעלת הפונקציה, קיבלנו משהו שמתקבל ב-L_2 אז לפני זה הוא התקבל ב-L_1 וגם להפך. למשל, אם L_2 לא עוצר על $f(x)$ אז גם L_1 לא יעצור.



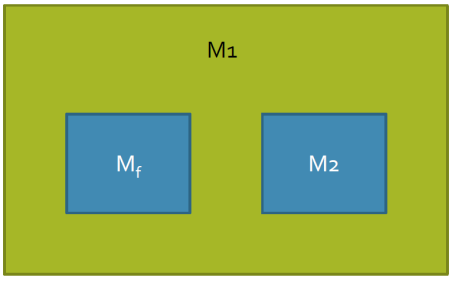
משפט הרדוקציה

אם $L_1 \leq L_2$ (אם L_1 **ניתנת לרדוקציה ל-**L_2) אז:

- אם $L_2 \in R$ אז גם $L_1 \in R$.
- אם $L_2 \in RE$ אז גם $L_1 \in RE$.
- אם $L_2 \in coRE$ אז גם $L_1 \in coRE$.

<u>רעיון ההוכחה (בציור)</u>:

- אם L_1 "קשה" כמו L_2, אז אנחנו יודעים לסווג את L_2 ואז גם את L_1 אנחנו נדע לסווג.
- אם אנחנו יודעים של-$L_2 \in RE$ יש מ"ט מקבלת אז גם ל-$L_1 \in RE$ אפשר יהיה לבנות מ"ט מקבלת.
- בציור: ההרכבה של המכונה M_2 (פונקציה ל-L_2) עם המכונה M_f (פונקצית רדוקציה) תיצור את המכונה M_1 שתתן פתרון עבור L_1.



הוכחת משפט הרדוקציה

אנחנו נוכיח עבור R, ועבור שתי המחלקות הנוספות - ההוכחה דומה.

<u>טענה</u>: אם $L_1 \leq L_2$ וגם $L_2 \in R$ אז גם $L_1 \in R$.

<u>הוכחה</u>: נניח ש: $L_1 \leq L_2$, אזי קיימת מכונת טיורינג M_f המחשבת את פונקציית הרדוקציה מ-L_1 ל-L_2.

ונניח שגם $L_2 \in R$, אזי קיימת מכונת טיורינג M_2 <u>המכריעה</u> את השפה L_2.

נתאר מכונה מכריעה M_1 לשפה L_1:

המכונה M_1 על קלט x:

1. מחשבת את $f(x)$ (על ידי סימלוץ של ריצת M_f על הקלט x).
2. מריצה את M_2 על $f(x)$ ועונה כמוה.

נכונות המכונה נובעת ישירות מהתכונות של פונקציית הרדוקציה.

- מכיוון שהפונקציה מלאה וניתנת לחישוב, השלב הראשון (חישוב הפונקציה) תמיד יעבור בהצלחה.
- הפונקציה תקפה:
  - $x \in L_1 \rightarrow f(x) \in L_2 \rightarrow M_2(f(x)) = 1 \rightarrow M_1(x) = 1 \rightarrow x \in L(M_1)$
  - $x \notin L_1 \rightarrow f(x) \notin L_2 \rightarrow M_2(f(x)) = 0 \rightarrow M_1(x) = 0 \rightarrow x \notin L(M_1)$
- התנאים בנוגע למתי המכונה M_1 עוצרת ומתי לא, זהים עבור המכונה M_2.

משפט הרדוקציה - ניסוח שקול (ושימושי יותר)

אם $L_1 \leq L_2$ (אם L_1 **ניתנת לרדוקציה ל**-L_2) אז:

- אם $L_1 \notin R$ אז גם $L_2 \notin R$.
- אם $L_1 \notin RE$ אז גם $L_2 \notin RE$.
- אם $L_1 \notin coRE$ אז גם $L_2 \notin coRE$.

<u>הערות</u>:

- נזכר בהוכחה בסוף של מצגת 2, למעשה הוכחנו שיש רדוקציה $L_D \leq HP$. ואז מכיוון ש: $L_D \notin R$ אזי לפי משפט הרדוקציה נובע שגם $HP \notin R$.
- בדרך כלל, כשנתעסק ברדוקציות, אנחנו נתעסק עם המשפט של הניסוח השקול.
  כלומר, אם $L_1 \leq L_2$ אז אפשר להגיד שהשפה L_1 יכולה להיות אחת מהשפות הבאות שנלמדו:
  $HP, L_D, L_U \notin R$ ואז גורר מכך ש: $L_2 \notin R$. כי אם L_2 "קשה" לפחות כמו L_1 אז בוודאות $L_2 \notin R$.
- כלומר, אם נסתכל על $L_1 \leq L_2$ אז אפשר להגיד שאם "השמאלי" לא ב-R אז גם "הימני" לא ב-R.
  וזה נכון גם עבור $RE,\ coRE$.

הוכחת $HP \notin R$ על ידי רדוקציה

נוכיח כעת שנית את אותה הטענה אבל בעזרת רדוקציה.

$$L_D = \{< M > \mid M\ accepts\ < M >\} \in RE$$

$$HP = \{< M, x > \mid M\ halts\ (stops)\ on\ x\} \in RE$$

נראה כי $HP \notin R$ אך הפעם נעשה זאת במבנה הוכחה של רדוקציה.

<u>טענה</u>: $HP \notin R$.

<u>הקדמה להוכחה</u>: נציג כאן מכונה M' שתייצג את HP ומכונה M שתייצג את L_D.

<u>הוכחה</u>: נוכיח ע"י בניית רדוקציה $L_D \leq HP$, ואז לפי משפט הרדוקציה נקבל שמכיוון ש: $L_D \notin R$ אז גם $HP \notin R$. פונקציית הרדוקציה: $f(< M >) =< M',\ < M >>$. (זה פונקציה $L_D \to HP$ כמו "המרה").

כאשר M' על קלט y:

- מסמלצת את ריצת M על y.
- אם M קיבלה אז גם M' מקבלת. (כי אמרנו ש: $L_D \in RE$ וב-RE כל מילה $x \in L_D$ הוא מתקבל <u>ועוצר</u>).

אם M דחתה אז M' נכנסת ללולאה אינסופית. (כי לפי הגדרת HP, המכונה שלה לא עוצרת [halts] לכל דחייה.
במילים אחרות, אם הקלט x לא התקבל ב-L_D אז הוא גם לא מתקבל ב-HP ולפי הגדרת השפה, המכונה של HP לא עוצרת).

כעת, צריך להוכיח שהפונקציה <u>מלאה, ניתנת לחישוב ותקפה</u> (מספיק להראות זאת בנוגע לקלט תקין):

- הפונקציה מלאה כיוון שמכל מכונה M שנתונה לנו, ניתן לבנות את המכונה M'.
- הפונקציה ניתנת לחישוב כיוון שלמדנו שכל מכונת טיורינג ניתנת לקידוד כמחרוזת.
- הפונקציה תקפה: צריך להראות ש: $\forall x \in \Sigma^* : x \in L_1 \Leftrightarrow f(x) \in L_2$.
  - אם $M\ (\ )\ M \in L_D$ מקבלת את $< M >$ ומקבלת $M'\ (\ )\ < M >$ גם עוצרת על $f(< M >) = < M', < M > > \in HP\ (\ )$.
  - אם $M\ (\ )\ M \notin L_D$ דוחה את $< M >$ או לא עוצרת $M'\ (\ )$ עוברת ללולאה אינסופית על $< M >$ או דוחה אותה $f(< M >) = < M', < M > > \notin HP\ (\ )$.

∎

$$L_D = \{< M > \mid M\ accepts\ < M >\} \in RE$$

$$HP = \{< M, x > \mid M\ halts\ (stops)\ on\ x\} \in RE$$

<u>אבחנה</u>: הפונקציה הזו מתאימה גם אם נחליף את השפה HP בשפה:

$$L_u = \{< M, x > \mid M\ accepts\ x\} \in RE$$

כלומר, $f(x) =< M',\ x >$. לכן מתקיים כי $L_D \leq L_u$. מכאן גם $L_u \notin R$.

- במילים אחרות, אם $L_D \leq HP$ אז אותה רדוקציה בדיוק תתאים גם ל-$L_D \leq L_u$ ואז $L_u \notin R$.

תכונות של יחס הרדוקציה

<u>תכונה 1</u>: אם $L_1 \leq L_2$ אז גם $\overline{L_1} \leq \overline{L_2}$.

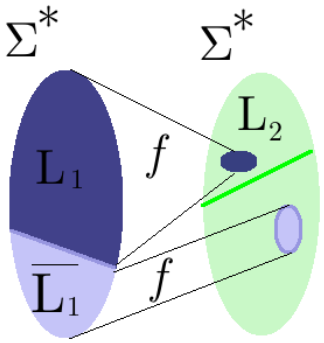
<u>הוכחה</u>: זוהי תוצאה ישירה של התקפות של הרדוקציה.

<u>שאלה</u>: האם אפשר להשתמש באותה הפונקציה f אבל עבור המשלים?

<u>תשובה</u>: כן. כי לפי התקפות של פונקצית הרדוקציה: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

לכן, אם $x \in L_1$ אז גם $f(x) \in L_2$ ולהפך, אם $x \notin L_1$ אז גם $f(x) \notin L_2$.

כלומר, אם $x \in \overline{L_1}$ אז גם $f(x) \in \overline{L_2}$. (המחשה בציור מלמטה):



<u>תכונה 2</u>: היחס $\leq$ הוא **רפלקסיבי**.

כלומר: לכל שפה L מתקיים $L \leq L$. במילים אחרות, כל פונקציה ניתנת לרדוקציה לעצמה.

<u>רעיון ההוכחה</u>: מתבצע בעזרת פונקציית <u>הזהות</u> $f(x) = x$.

<u>תכונה 3</u>: היחס $\leq$ הוא **טרנזיטיבי**.

כלומר: אם $L_1 \leq L_2$ וגם $L_2 \leq L_3$ אז גם $L_1 \leq L_3$.

<u>רעיון ההוכחה</u>: הרכבת פונקציות: אם $f(L_1) \to L_2$ וגם $g(L_2) \to L_3$ אז $g(f(L_1)) \to L_3$.

<u>תכונה 4</u>: היחס $\leq$ הוא **<u>לא</u> סימטרי**.

כלומר: אם $L_1 \leq L_2$ אז <u>לא בהכרח</u> מתקיים ש: $L_2 \leq L_1$.

<u>דוגמה</u>: ניקח את השפות HP ו- Σ^*.

- <u>מצד אחד</u> - אפשר לחשוב על רדוקציה $HP \leq \Sigma^*$, למשל $f(x) = < M_{STAM}, x >$.

  תזכורת - המכונה M_{STAM} מעבירה מיד קידוד לא תקין למצב q_{accept}.

  כל מילה $x \in \Sigma^*$ תמיד מתקבלת. לכן, נבחר במכונה M_{STAM} שמקבלת כל x.

- <u>מצד שני</u> - $\Sigma^* \leq HP$ לא מתקיים, כי אז לפי משפט הרדוקציה היינו מקבלים כי גם $HP \in R$.

  אבל ב-HP יש קלטים שלא עוצרים, וב-Σ^* הכל מתקבל בשפה כי זה כל x (ניתן להכרעה).

תרגיל 1 - שאלה לדוגמה

נתונה השפה: $L = \{< M > \ : \ |L(M)| \geq 4 \}$.

<u>סעיף א'</u>: קבעו האם $L \in RE$? הוכיחו את תשובתכם.

<u>פתרון סעיף א'</u>: אכן השפה $L \in RE$. נוכיח ע"י תיאור מ"ט <u>מקבלת</u> לשפה L.

נצטרך פה הרצה מבוקרת "חזקה", שיכולה (תיאורטית) לעבור על כל המילים ב-Σ^*.

	<u>המכונה U על קלט</u> $< M >$:
1	נשתמש בסידור לקסיקוגרפי (אינסופי) של Σ^* כך: $w_1, w_2, w_3, \ldots$.
2	אתחל מונה $counter = 0$. בכל שלב אם $counter = 4$ אז עצור <u>וקבל</u>.
3	לכל $i = 1, 2, 3, \ldots$ בצע:
4	לכל $1 \leq j \leq i$ בצע:
5	הרץ את המכונה M על המילה w_j למשך i צעדים. (כלומר נריץ $M(w_j[1 \ldots i])$).
6	אם M קיבלה את המילה w_j אז הוסף למונה: $counter = counter + 1$.
7	אפס את המונה $counter = 0$.

נוכיח שהמכונה U <u>מקבלת</u> את השפה L:

- אם $< M > \in L$ אז קיימות 4 במילים w_i, w_j, w_k, w_r המתקבלות בכמות הצעדים <u>הקטנה</u> ביותר - t ונניח בה"כ ש: $i, j, k, r < t$. אזי באיטרציה t של האלגוריתם, נספק לכל מילה t צעדים עד ש-M יריץ את המילה w_t. באיטרציה זו, נקבל את w_i, w_j, w_k, w_r ולכן $< M > \in L(U)$.
- אם $< M > \notin L$ אז לא קיימות לפחות 4 מילים כאלה, אנחנו נמשיך לרוץ בלולאה ונחפש את המילים האלה למרות שהם לא קיימות - לכן אנחנו לא נעצור לעולם ונרוץ לאינסוף. ולכן $< M > \notin L(U)$.

הוכחנו ש: $L(U) = L$ וכן המכונה U <u>מקבלת</u> את השפה L כנדרש.

∎

<u>סעיף ב'</u>: קבעו האם $L \in R$? הוכיחו את תשובתכם.

<u>אינטואיציה</u>: הרגע הוכחנו ש: $L \in RE$ והראנו באלגוריתם בהרצה מבוקרת כי אם $< M > \notin L$ אז לא נעצור לעולם. לכן, לא נוכל לבנות מכונה שתכריע את השפה הזאת.

<u>פתרון סעיף ב'</u>: השפה L איננה ב-R. נוכיח זאת ע"י רדוקציה $HP \leq L$.

פונקצית הרדוקציה היא: $f(< M, x >) = < M_x >$.

כאשר M_x על קלט y אז:

- מריצה את M על x (מתעלמת מהקלט y).

- M_x מקבלת את y.

נכונות הפונקציה:

- הפונקציה מלאה - לכל קידוד של מכונה וקלט נתונים, ניתן לבנות את המכונה M_x.
- הפונקציה ניתנת לחישוב - כל מ"ט ניתנת לקידוד כמחרוזת.
- הפונקציה תקפה - צריך להראות ש: $\forall x \in \Sigma^* : x \in L_1 \Leftrightarrow f(x) \in L_2$.
  - אם $< M, x > \in HP$ (M עוצרת על x) גם M_x ( ) עוצרת על y ומקבלת $f(< M, x >) =< M_x > \in L$ ( ).
  - אם $< M, x > \notin HP$ (M לעולם לא עוצרת על x ( )) גם M_x ( ) לעולם לא עוצרת על y ולכן $f(< M, x >) =< M_x > \notin L$ ( ).

∎

<u>הסבר על הרדוקציה הזאת</u>: לפי משפט הרדוקציה (השקול), עבור $HP \leq L$ אז אם $HP \notin R$ גם $L \notin R$.
פונקציית הרדוקציה לוקחת קלט שהיא מילה $< M, x >$ מהשפה HP, ומחזירה פלט שהיא מילה $< M_x >$ משפה L. אם M עצרה על x, אז M_x צריכה לקבל לפחות 4 מילים. לכל קלט y ב-M_x, המכונה תמיד תקבל אותה, למה? כי M_x "מפעילה את M על x" ובגלל ש-"מפעילה את M על x" תמיד עוצר (כי היא שייכת ל-HP) אז תמיד נוכל להמשיך לשלב הבא באלגוריתם שזה לקבל את y תמיד ללא תנאים.
ואם y תמיד תתקבל לא משנה איזה y זה, אז זה אומר ש: M_x מקבל כל דבר - כלומר $\left|L(M_x)\right| = \infty$.
לכן, יש בהכרח לפחות 4 מילים בשפת המכונה ואם כך $< M_x > \in L$.

תרגיל 2 - שאלה לדוגמה

נתונה השפה: $L_2 = \{< M > \mid M \; stops \; for \; every \; input \; x \in \Sigma^*\}$.
נשים לב שהרדוקציה האחרונה שראינו, מתאימה גם לשפה הזו, כי אם M עוצרת על כל x אז היא בוודאי ש: $L_2 \notin R$. בנוסף, $L_2 \notin coRE$, נחזור לזה בהמשך.
השפה איננה ב-R. נוכיח זאת ע"י רדוקציה $HP \leq L_2$.
פונקצית הרדוקציה היא: $f(< M, x >) =< M_x >$.
כאשר M_x על קלט y אז:

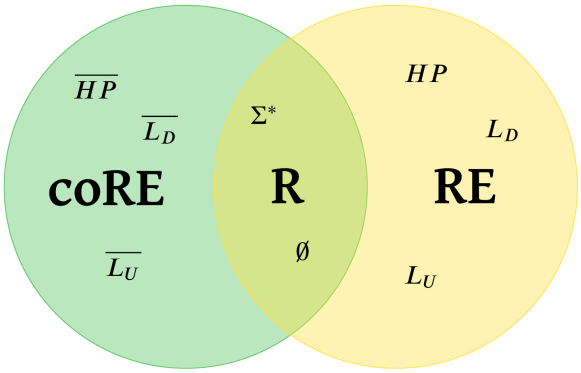
- מריצה את M על x (מתעלמת מהקלט y).
- מקבלת (את y).

נכונות הפונקציה דומה להוכחה ברדוקציה שעשינו בשאלה הקודמת.

תמונת העולם עד כה

עד עכשיו, הוכחנו ולמדנו על כל מה שמופיע בציור משמאל, כולל השפות המשלימות ב-$coRE$.

מה לגבי שפות שלא שייכות ל-RE וגם לא שייכות ל-$coRE$? נראה כאלה עכשיו.



הקדמה - רוב השפות לא שייכות ל-RE ולא ל-$coRE$

- כל מכונת טיורינג מורכבת משביעייה $M = \left(Q, q_0, F, \Gamma, \Sigma, \bar{b}, \delta\right)$.
- כל איבר איבר בשביעייה מכיל קבוצה שגודלה הוא בן מניה.
- מ"ט מורכבת ממכפלה קרטזית של 7 קבוצות בנות מניה.
- לכן יש מספר בן מניה של שפות ב-RE.
- אבל, יש מספר לא בן מניה של שפות (הוכחנו בקורס אוטומטים).

תרגיל 3

נציג כאן דוגמה לשפה ששייכת ל- $\overline{RE \cup coRE}$ (כלומר שפה שלא ב-RE ולא ב-$coRE$).

נתונה השפה הבאה: $L_\infty = \{< M > : |L(M)| = \infty\}$.

<u>אינטואיציה</u>: גם אם נצליח לוודא שהמכונה מקבלת המון קלטים, כל כמות שנצליח לבדוק היא בהכרח סופית. לדוגמה, לא נוכל לבצע הרצה מבוקרת על אינסוף מילים, כי הרצה מבוקרת תמיד תעצור לפי הגדרתה. בכל איטרציה, היא עוצרת על כמות סופית של מילים ותבדוק אותם.

<u>טענה</u>: $L_\infty \in \overline{RE \cup coRE}$.

<u>מבוא</u>: ההוכחה תהיה בשני שלבים:

1. $HP \leq L_\infty$ - ממשפט הרדוקציה נובע ש: $L_\infty \notin coRE$.
2. $\overline{HP} \leq L_\infty$ - ממשפט הרדוקציה נובע ש: $L_\infty \notin RE$.

<u>הוכחה - חלק (1)</u>: נוכיח זאת ע"י רדוקציה $HP \leq L_\infty$.

פונקצית הרדוקציה היא: $f(< M, x >) =< M_x >$.

כאשר M_x על קלט y אז:

- מריצה את M על x (מתעלמת מהקלט y).
- מקבלת (את y).

נכונות הפונקציה דומה כמו שעשינו בתרגילים קודמים.

<u>הוכחה - חלק (2)</u>: נוכיח זאת ע"י רדוקציה $\overline{HP} \leq L_\infty$.

(כלומר, נרצה להוכיח שאם M לא עוצרת על x אז נרצה שהשפה תהיה אינסופית).

פונקצית הרדוקציה היא: $f(< M, x >) =< M'_x >$.

כאשר M'_x על קלט $y \in \Sigma^*$ אז:

1. מריצה את M על x למשך $|y|$ צעדים.
2. אם M לא עצרה על x בשלב 1 - <u>קבל</u>.
3. אם M עצרה על x - <u>דחה</u>.

נכונות הפונקציה דומה כמו שעשינו בתרגילים קודמים.

<u>תקפות הרדוקציה</u>:

1. אם $< M, x > \in \overline{HP}$ ( ) M ( ) לא עוצרת על x ( ) לא עצרה גם בתוך $|y|$ צעדים עבור כל y ( ) M'_x תקבל כל y ( ) $L(M'_x) = \Sigma^*$ ( ) $\left|L(M'_x)\right| = \infty$ ( ) $< M'_x > \in L_\infty$ ( ).
2. אם $< M, x > \notin \overline{HP}$ ( ) M ( ) עוצרת על x לאחר מספר סופי k של צעדי חישוב ( ) עבור כל y המקיים $|y| > k$ אז M'_x תדחה את y ( ) M' תקבל רק מילים y כך ש: $|y| < k$ ( ) $\left|L(M'_x)\right| < \infty$ ( ) $< M'_x > \notin L_\infty$.

∎

<u>הערה - חלק (2)</u>:

- $\overline{HP} = \{< M, x > \mid M\ doesn't\ stop\ on\ x\}$.
- יש לנו אינסוף מילים, כל מילה בפני עצמה בכמות סופית של תווים (כל מילה היא באורך סופי).
- כל עוד אנחנו במילים קטנות y אז השפה עדיין לא אינסופית, אבל כשאנחנו מגיעים למילים ארוכות מאוד אז אנחנו נותנים ל-M כמה צעדים שהיא צריכה.
- לכן, אם M לא עצרה, אז היא צריכה הרבה יותר ממספר הצעדים שהיא קיבלה בזכות $|y|$שעליה היא מסתמכת, כלומר זה אינסופי ולכן נעצור ונקבל.
- אם M עצרה אז הגענו "לחסם" מסוים לאחר t צעדים ולכן לכל y המקיים $|y| \geq t$ אז $y \notin < M'_x >$ כלומר, זה לא אינסופי ונדחה.

המשך תרגיל 2

נחזור לשפה $L_2 = \{< M > \mid M\ stops\ for\ every\ input\ x \in \Sigma^*\}$. ראינו ש: $L_2 \notin R$.

מה לגבי שייכות ל-RE?: אפשר לשנות מעט את הרדוקציה שראינו $\overline{HP} \leq L_\infty$ כדי להתאים לשפה L_2.

<u>טענה</u>: $L_2 \notin RE$.

<u>הוכחה</u>: נראה כי $\overline{HP} \leq L_2$. פונקצית הרדוקציה היא: $f(< M, x >) =< M'_x >$.

כאשר M'_x על קלט w אז:

1. מריצה את M על x למשך $|w|$ צעדים.

2. אם M לא עצרה על x בשלב 1 - קבל.
3. אם M עצרה על x - לולאה אינסופית.

נכונות הפונקציה של מלאה וניתנת לחישוב נעשה דומה כמו שעשינו בתרגילים קודמים.

תקפות הרדוקציה:

1. אם $< M, x > \in \overline{HP}$ (M לא עוצרת על x) M'_x מקבל את w (M'_x עוצרת לכל w) $M'_x \in L_2$.
2. אם $< M, x > \notin \overline{HP}$ (M עוצרת על x לאחר מספר סופי t של צעדי חישוב) M'_x תיכנס ללולאה אינסופית עבור כל w כך ש: $|w| \geq t$ (M'_x לא עוצרת לכל w) $M'_x \notin L_2$.

הסבר להמשך תרגיל 2: נשים לב ש: $\overline{HP} = \{< M, x > \mid M\ doesn't\ stop\ on\ x\}$.

נזכור שלא באמת מעניין אותנו התוכן של w, אלא רק צריך להגיב בהתאם לעניין השייכות של $\overline{HP}$:

1. אם M לא עצרה על x, אז זה מעולה, כי זה עונה על דרישת השפה $\overline{HP}$, כלומר $< M, x > \in \overline{HP}$ ולכן גם המכונה M'_x של L_2 חייבת להגיב בהתאם למקרה של $M'_x > \in L_2$ ולכן, תקבל את w.
2. אם M עצרה על x, אז זה לא עונה על דרישת השפה $\overline{HP}$, כלומר $< M, x > \notin \overline{HP}$.
   מתי מילה לא שייכת לשפה L_2? - כאשר M לא עוצר עבור כל $x \in \Sigma^*$.
   לכן המכונה M'_x של L_2 חייבת לא לעצור לעולם על כל x.

תרגיל 4

נתונה השפה $L_{\Sigma^*} = \{< M > \mid L(M) = \Sigma^*\}$.

טענה: $L_{\Sigma^*} \in \overline{RE \cup coRE}$.

קצת סדר בראש: השפה $R \in \Sigma^*$ כי ניתן להכריע אותה מיידית - נעבור על כל הסרט של המכונה, אם כל התווים מוכרים לנו ב-Σ אז נקבל את המילה בסוף הסרט, אם לא אז נדחה ישר. כאן המקרה הוא שונה ויכול לבלבל, השפה L_{Σ^*} היא שפת כל המכונות המקבלות את השפה Σ^*. לכן, שפה L_{Σ^*} היא הרבה יותר "קשה" מ-R ולכן היא מחוץ ל"מגרש המשחקים" שהתעסקנו בו עד עכשיו, כלומר היא $L_{\Sigma^*} \in \overline{RE \cup coRE}$.

אבחנה: שתי פונקציות הרדוקציה שראינו עבור $HP \leq L_\infty$, $\overline{HP} \leq L_\infty$ מתאימות גם עבור השפה L_{Σ^*}.

הוכחה חלק 1: אנו יודעים כי $HP \notin coRE$, נוכיח ברדוקציה $HP \leq L_{\Sigma^*}$ וכך נראה כי $L_{\Sigma^*} \notin coRE$.

פונקציית הרדוקציה מוגדרת כך: $f(< M, x >) = < M_x >$.

M_x על קלט w אז:

- מריצה את M על x.
- מקבל את w.

נוכיח את נכונות הגדרת הפונקציה.

1. הפונקציה מלאה: לכל קידוד של מכונה וקלט נתונים, ניתן לבנות את המכונה M_x.
2. הפונקציה ניתנת לחישוב: כי קיים קידוד למחרוזת לכל מ"ט.
3. הפונקציה פתירה:
   a. אם $< M, x > \in HP$ ← M ← עוצר על x ← לפי הבנייה, M_x מקבל את w ← $M_x \in L_{\Sigma^*}$.
   b. אם $< M, x > \notin HP$ ← M ← לא עוצר על x ← לפי הבנייה, M_x רץ לאינסוף ← $M_x \notin L_{\Sigma^*}$.

<u>הוכחה חלק 2</u>: אנו יודעים כי $\overline{HP} \notin RE$. נוכיח ברדוקציה $\overline{HP} \leq L_{\Sigma^*}$ וכך נקבל $L_{\Sigma^*} \notin RE$.

פונקציית הרדוקציה מוגדרת כך: $f(< M, x >) =< M_x >$.

M_x על קלט w תבצע:

- M תפעל על x למשך $|w|$ צעדים.
  - אם M לא עצרה על x - <u>קבל</u>.
  - אם M עצרה על x - <u>רוץ לאינסוף</u>.

נוכיח את נכונות פונקצית הרדוקציה.

1. הפונקציה מלאה: לכל קידוד של מכונה וקלט נתונים, ניתן לבנות את המכונה M_x.
2. הפונקציה ניתנת לחישוב: קיים קידוד כמחרוזת לכל מ"ט.
3. הפונקציה פתירה:
   a. אם $< M, x > \in \overline{HP}$ ← M ← לא עצרה על x ← M_x מקבל כל קלט באורך $|w|$ ← $M_x \in L_{\Sigma^*}$.
   b. אם $< M, x > \notin \overline{HP}$ ← M ← עצרה על x לאחר t צעדים ← M_x רץ לאינסוף החל מהצעד ה-t ← מילה לא התקבלה כל מילה חייבת להיות ב-Σ^* ← $M_x \notin L_{\Sigma^*}$.

תרגיל 5

נתונה השפה $L_{eq} = \{< M_1, M_2 > \mid L(M_1) = L(M_2)\}$.

<u>טענה</u>: $L_{eq} \in \overline{RE \cup coRE}$.

<u>אינטואיציה</u>: אם נמצא מילה x כלשהי ש-M_1 מקבלת ו-M_2 דוחה אז $L(M_1) \neq L(M_2)$, כלומר זיהינו מקרה של קלט שהוא לא בשפה, אז אפשר לחשוב תחילה כי $L_{eq} \in coRE$. מה יכול להיות בעייתי? - לא מובטח לנו ש M_2 תדחה את המילה - יכול להיות שתרוץ לאינסוף על x ולא נוכל לזהות מקרים: $L(M_1) = L(M_2)$. לכן, יש לנו כאן בעיה "קשה" יותר משפות ב-$coRE$ ולכן $L_{eq} \in \overline{RE \cup coRE}$.

<u>מבוא להוכחה</u>: אפשר להוכיח ע"י שתי רדוקציות, כמו קודם. אבל נעשה קיצור דרך:
אפשר להוכיח ע"י רדוקציה משפה שאנחנו כבר יודעים עליה שהיא ב-$\overline{RE \cup coRE}$.

<u>תזכורת</u> - M_{STAM} היא מכונה המקבלת כל קלט.

$$L_{\Sigma^*} = \{< M > \mid L(M) = \Sigma^*\}$$
$$L_{eq} = \{< M_1, M_2 > \mid L(M_1) = L(M_2)\}$$

<u>הוכחה</u>: נוכיח בעזרת הרדוקציה $L_{\Sigma^*} \leq L_{eq}$ שמתקיים $L_{eq} \in \overline{RE \cup coRE}$.

פונקציית הרדוקציה: $f(< M >) = (< M, M_{STAM} >)$.

<u>הערה</u>: למה אפשר להשתמש ב-M_{STAM}? כי $< M_{STAM} > \in L_{\Sigma^*}$ כי באמת מתקיים $L(M_{STAM}) = \Sigma^*$ בגלל שהיא מקבלת כל קלט, כלומר כל מילה שזה בדיוק ההגדרה ל-Σ^*.

נוכיח את נכונות פונקציית הרדוקציה:

1. הפונקציה מלאה: לכל מכונה M אפשר להצמיד מכונה נוספת M_{STAM}.
2. הפונקציה ניתנת לחישוב: כי אפשר ליצור קידוד כמחרוזת לכל מ"ט.
3. הפונקציה פתירה:
   a. אם $< M > \in L_{\Sigma^*}$ $\leftarrow$ לפי הגדרת השפה $L(M) = \Sigma^*$ $\leftarrow L(M) = L(M_{STAM})$ $\leftarrow$ $< M, M_{STAM} > \in L_{eq}$.
   b. אם $< M > \notin L_{\Sigma^*}$ $\leftarrow$ לפי הגדרת השפה $L(M) \neq \Sigma^*$ $\leftarrow L(M) \neq L(M_{STAM})$ $\leftarrow$ $< M, M_{STAM} > \notin L_{eq}$.

תרגול 3 - שאלה 1

עבור השפה הבאה $L_\varepsilon = \{< M > \mid M \ accepts \ \varepsilon\}$ הוכיחו כי השפה איננה ב-R.

<u>פתרון</u>:

- <u>אינטואיציה</u>: ייתכן שעבור מ"ט M לא תעצור לעולם על קלט ε ולכן $L \in RE$.
- <u>תיאור הרדוקציה</u>: נראה רדוקציה $HP \leq L_\varepsilon$. פונקציית הרדוקציה: $f(< M, x >) =< M' >$ כך ש-M' על כל קלט w:
  1. מסמלצת את ריצת M על x.
  2. אם M עצרה - M' <u>תקבל</u>.
- <u>נכונות פונקציית הרדוקציה</u>:
  - הפונקציה מלאה: כי M' מוגדרת לכל M ו-x.
  - הפונקציה ניתנת לחישוב: לכל מכונה קיים קידוד למחרוזת.
  - הפונקציה תקפה:
    - אם $< M, x > \in HP$ $\leftarrow$ M עצר על x $\leftarrow$ M' מקבל את w $\leftarrow L(M') = \Sigma^*$ $\leftarrow \varepsilon \in L(M')$ $\leftarrow M' \in L_\varepsilon$.

- אם $M \leftarrow < M, x > \notin HP$ לא עצר על $x \leftarrow M'$ רץ לאינסוף $\leftarrow L(M') = \emptyset \leftarrow \varepsilon \notin L(M') \leftarrow L_\varepsilon \notin M'$.

- <u>סיכום</u>: הראינו כי HP ניתנת לרדוקציה ל-L_ε ידוע לנו כי $HP \notin R$ ולכן $L_\varepsilon \notin R$ כנדרש.

תרגול 3 - שאלה 2

נתונה השפה $L = \{< M > : M\ on\ \varepsilon\ iterate\ 3\ consecutive\ steps\ to\ the\ right\}$.
כלומר, $< M > \in L$ אם M בריצתה על ε מבצעת 3 צעדים רצופים ימינה.

<u>סעיף א'</u>: האם השפה ב-RE? - הוכיחו.

<u>סעיף ב'</u>: האם השפה ב-R? - הוכיחו.

<u>פתרון סעיף א'</u>: השפה אכן ב-RE. לפי הגדרת RE קיים מכונה M המקבלת את השפה L. אכן אפשר לבנות מכונה M שכזאת, נבנה מונה (counter) שסופר את מספר הצעדים רצוף ימינה, אם הגענו ל-3 צעדים אז נקבל. אם זזנו שמאלה או נשאר במקום אז נאפס את המונה. כדי לתת תשובה פורמלית לסעיף זה, צריך להוכיח ע"י בניית מכונה. לא נוכיח כאן כי זה הרבה עבודה אבל ניתן רעיון כללי:

1. אם בתחילת האלגוריתם, הראש קרא מתא 1 את $\bar{b}$ אז אכן מדובר במילה הריקה.
2. נשתמש בסרט נוסף שיכיל את המונה (counter) ובו נעדכן בהתאם כמה צעדים רצופים ימינה עשינו.
3. נזוז ימינה R, אם התא הבא גם $\bar{b}$ (הוא כזה כי המילה ריקה) אז נעשה $counter\ ++$ בסרט השני.
4. אם נזוז שמאלה L או נשאר במקום S אז נאפס את המונה $counter = 0$ בסרט השני.
5. אם $counter == 3$ אז נעבור למצב מקבל q_{accept}.

<u>פתרון סעיף ב'</u>:

1. <u>אינטואיציה</u>: קל לראות ש: $L \in RE$, כאשר נסמלץ ריצה של M על ε, נוכל לזהות שהיא מבצעת 3 צעדים רצופים ימינה. אבל אם יש לדוגמה מ"ט כלשהי M' כלשהי שעבור $x = \varepsilon$ רצה לאינסוף ועבור $x \neq \varepsilon$ עוצרת ומקבלת, אז לא נוכל להכריע במקרה זה. כלומר אנחנו לא נוכל להגיד <u>לא</u> כי המכונה תקועה בלולאה אינסופית ולכן לא נוכל לזהות מה הפלט הנכון.
   לכן, נוכיח שלא שייכת ל-R באמצעות השפה L_ε והוכחנו כבר עליה כי $L_\varepsilon \notin R$.
   נוכיח שמתקיים: $L \leq L_\varepsilon$. כלומר נוכיח כי אם $L_\varepsilon \notin R$ אז מהרדוקציה נקבל גם כי $L \notin R$.
2. <u>תיאור הרדוקציה</u>: נראה רדוקציה: $L \leq L_\varepsilon$.
   פונקצית הרדוקציה תהיה: $f(< M >) =< M' >$.
   כאשר M' על קלט x:
   - מסמלצת את M על ε, כאשר <u>בכל צעד</u>:
     - אם M' עשתה צעד R, המכונה M מבצעת צעד S שלא משנה כלום.
     - אם M קיבלה, אז M' הולכת שלושה צעדים ימינה <u>ומקבלת</u>.
     - אם M דחתה, אז M' <u>דוחה</u>.

<u>הערה</u>: אנחנו נאפשר ל-M' לזוז 3 צעדים ימינה רק כאשר M תקבל את ε. בכל צעד של M, אם נזוז ימינה, אז M' תישאר במקום S ולאחר מכן תזוז ימינה R (כלומר ניצור איזשהו מצב שיגיד לי "תשאר צעד במקום ואז תזוז ימינה"). הסתכלות אחרת היא להגיד שM' יכולה לעשות RRR רק כאשר

M תקבל את ε, בשביל למנוע מצב ש-M' תעשה RRR כשלא באמת צריך, אז נגיד שלכל צעד ימינה, נעמוד במקום ואז נזוז ימינה, ואז יהיה לנו רצף צעדים של $SRSRSRSR$ וככה אנחנו מתחייבים שלא יהיה לנו לפחות 3 ימינה רצופים כנדרש.

3. נכונות פונקציית הרדוקציה:
   a. הפונקציה מלאה: כי M' מוגדרת לכל M.
   b. הפונקציה ניתנת לחישוב: כי קיים קידוד לכל מ"ט.
   c. הפונקציה פתירה:
      i. $< M > \in L_\varepsilon \leftarrow$ תעצור ותקבל את $\varepsilon \leftarrow M'$ בריצתה על ε תלך שלושה צעדים רצופים ימינה $\leftarrow < M' > \in L$.
      ii. $< M > \notin L_\varepsilon \leftarrow$ תעצור ותדחה או לא תעצור $\leftarrow M'$ בריצתה על ε לא תלך שלושה צעדים רצופים ימינה ואפילו אינסוף (משום שקטענו עם S) $\leftarrow < M' > \notin L$.
4. סיכום: הראינו כי L_ε ניתנת לרדוקציה ל-L ידוע לנו כי $L_\varepsilon \notin R$ ולכן $L \notin R$ כנדרש.

תרגול 3 - שאלה 3

נתונה השפה הבאה: $L = \{< M > : M(001) = 001\}$.

סעיף א': האם השפה הנתונה ב-RE?

סעיף ב': האם השפה הנתונה ב-R?

פתרון סעיף א': אם M לא עוצרת על 001 אז בפרט המילה הזאת לא בשפה והמכונה לא קולטת את המילה הזאת. אבל לפי RE אם היא לא בשפה אז מותר לנו שהמכונה לא תעצור וזה בסדר... כלומר השפה הנתונה כן ב-RE. כדי להוכיח ששפה שייכת למחלקה, צריך לבנות מ"ט שמקבלת את השפה. (הוכחה בבית).

פתרון סעיף ב':

1. אינטואיציה: קל לראות ש: $L \in RE$ (הראינו בסעיף א'), אבל ייתכן שקיים מכונה M כלשהי כך שבהינתן הקלט 001, היא לא תדע איך לעבוד איתה ולכן תיכנס ללולאה אינסופית - כלומר היא לא תוכל להכריע את דרישת השפה L. לכן, השפה $L \notin R$. נרצה להוכיח ש$L \notin R$ ולכן נשתמש ברדוקציה. אנחנו צריכים לחשוב על שפה אחרת שהוכחנו עליה כבר שהיא לא ב-R אבל כן ב-RE.
2. תיאור הרדוקציה: נראה רדוקציה $HP \leq L$.
   פונקציית הרדוקציה היא $f(< M, x >) =< M_x >$.
   כאשר M_x על קלט w:
   - תסמלץ את M על x.
   - אז קבל (מחזירה את w).

   הערה: אם M תעצור על x, אז נחזיר את w. ה-w יכול להיות כל מיני מחרוזות, $w = 011$ או $w = 110$ או $w = 001$ וכו'... לכן, אם M לא עצרה על x, אז לא נחזיר את w ואז $< M >$ לא תהיה בשפה וזה בסדר. ואם M כן עצרה על x אז נקבל את $< M >$ לשפה.
3. נכונות פונקציית הרדוקציה:
   a. הפונקציה מלאה: לכל M ו-x, ניתן ליצור M_x.
   b. הפונקציה ניתנת לחישוב: לכל מכונה קיים קידוד.
   c. הפונקציה פתירה:

i. אם $M \leftarrow < M, x > \in HP$ עוצרת על $M_x \leftarrow x$ גם עוצרת ומחזירה לכל $w \leftarrow$ בפרט אם $w = 001$ כך ש: $M_x(001) = 001 \leftarrow L$ $< M_x > \in$.

ii. אם $M \leftarrow < M, x > \notin HP$ לא עוצרת על $M_x \leftarrow x$ רצה לאינסוף, כלומר לא עוצרת לכל $w \leftarrow$ בפרט $M_x(001)$ לא מוגדרת$\leftarrow L$ $< M_x > \notin$.

4. <u>סיכום</u>: הראינו כי HP ניתנת לרדוקציה ל-L ידוע לנו כי $HP \notin R$ ולכן $L \notin R$ כנדרש.

תרגול 3 - שאלה 4

נתונה השפה $L_{=3} = \{< M >: |L(M)| = 3\}$.

<u>סעיף א'</u>: האם השפה ב-R?

<u>סעיף ב'</u>: האם השפה ב-RE?

<u>פתרון א'</u>: אינטואיציה: לא, כי לפי משפט רייס (התרגיל הוצג אחרי שנלמד הנושא הזה) אם תכונת השפה קשורה לתנאי המכונה אז היא לא ב-R.

<u>פתרון ב'</u>:

1. <u>אינטואיציה</u>: נראה ש: $\overline{HP} \leq L_{=3}$ וממנה נסיק כי $L_{=3} \notin RE$. כרגיל, אנו מתחילים עם קלט $< M, x >$ ורוצים לבנות ממנו מכונה M_x כך ש: $< M_x > \in L_{=3}$ רק אם M <u>לא</u> עוצרת על x. הרעיון הוא לקבל מייד 3 מילים כלשהן, ואם M עוצרת על x אז לקבל את כל שאר המילים האפשריות. (אנחנו רוצים "להיכשל").
2. <u>תיאור הרדוקציה</u>: נראה ברדוקציה $\overline{HP} \leq L_{=3}$.

   פונקציית הרדוקציה: $f(< M, x >) =< M_x >$.

   כאשר M_x על $w \in \Sigma^*$ פועלת כך:

   1. אם $w \in \{\varepsilon, 0, 1\}$ אז M_x מקבלת.
   2. M_x תסמלץ את M על x.
   3. אם M עצרה, M_x מקבלת.

   <u>הסבר</u>: אם M <u>לא</u> עוצרת על x אז $< M, x > \in \bar{HP}$ ולפי ההגדרה של רדוקציה, אנחנו צריכים לחקות את השפה השניה, ולכן אנחנו חייבים שגם $< M_x > \in L_{=3}$. איך נעשה את זה? אם M לא עוצר על ה-x הזה, אז אנחנו צריכים 3 מילים שיתקבלו בשפת המכונה M_x, בה"כ נבחר את המילים $\{\varepsilon, 0, 1\}$. איך זה עובד בפועל? M_x תעבור ותפעל על כל $w \in \Sigma^*$, ואם M לא עוצר על x אז הוא בחיים לא יעבור לשלב (3), וזה נכון לכל w חוץ מ-$w \in \{\varepsilon, 0, 1\}$ שכן יכולות להתקבל כבר בשלב (1). ככה פתרנו שאם M לא עוצר על x אז M_x מקבל רק 3 מילים בלבד כנדרש.

   מצד שני, אם M עוצרת על x, אז $< M, x > \notin \bar{HP}$ זה אומר שאנחנו חייבים שגם $< M_x > \notin L_{=3}$. איך נעשה את זה? - אם M עוצרת על x אז לכל $w \in \Sigma^*$ תתקבל ולכן $L(M_x) = \Sigma^*$ ובהכרח השפה לא בגודל 3 כנדרש.

3. <u>נכונות פונקציית הרדוקציה</u>:
   a. הפונקציה מלאה: לכל M ו-x קיים M_x.
   b. הפונקציה ניתנת לחישוב: כל מ"ט ניתן לקידוד.

c. הפונקציה פתירה:

i. אם $M \leftarrow < M, x > \in \overline{HP}$ לא עוצרת על $x \leftarrow$ לכל $w \in \Sigma^*$, M_x מקבל רק את $w \leftarrow w \in \{\varepsilon, 0, 1\}$ ואכן בגודל 3 $\leftarrow L(M_x) = \{\varepsilon, 0, 1\}$. $< M_x > \in L_{=3}$

ii. אם $M \leftarrow < M, x > \notin \overline{HP}$ עוצרת על $x \leftarrow M_x$ מקבל כל $w \in \Sigma^* \leftarrow$ ובהכרח לא בגודל 3 $\leftarrow L(M_x) = \Sigma^*$. $< M_x > \notin L_{=3}$

4. <u>סיכום</u>: הראינו כי $\overline{HP}$ ניתנת לרדוקציה ל-$L_{=3}$ ידוע לנו כי $\overline{HP} \notin RE$ ולכן $L_{=3} \notin RE$ כנדרש.

תרגול 3 - שאלה 5 (שאלת מבחן)

נתונה השפה: $L = \{< M_1, M_2 > : \left|L(M_1) \cap L(M_2)\right| = \infty\}$. האם $L \in R$? האם $L \in RE$ - הוכיחו.

<u>הוכחה</u>: נוכיח כי $L \notin R$ וגם $L \notin RE$.

- <u>תיאור הרדוקציה</u>: נראה רדוקציה $\overline{HP} \leq L$.
  - פונקצית הרדוקציה: $f(< M, x >) =< M_x, M_{STAM} >$.
  - עבור M_x על קלט w בצע:
    - סמלץ את M על x למשך $|w|$ צעדים.
    - אם M עצרה - <u>דחה</u>.
    - אחרת לאחר ה-$|w|$ צעדים <u>מקבלת</u>.
  - עבור M_{STAM} מקבלת כל קלט.
- <u>נכונות פונקצית הרדוקציה</u>:
  - הפונקציה מלאה: לכל קלט (אם $< M, x >$ קידוד חוקי) קיים פלט.
  - הפונקציה ניתנת לחישוב: מדובר בפעולת קומפילציה פשוטה.
  - הפונקציה תקפה:
    - $< M, x > \in \overline{HP} \leftarrow M \leftarrow$ לא עוצרת לכל $x \leftarrow M_x$ מקבל כל $w \leftarrow L(M_x) = \Sigma^* \leftarrow$ גם המכונה M_{STAM} מקבלת כל קלט $\leftarrow \left|L(M_x) \cap L(M_{STAM})\right| = \left|\Sigma^* \cap \Sigma^*\right| = \left|\Sigma^*\right| = \infty \leftarrow L \in < M_x, M_{STAM} >$.
    - $< M, x > \notin \overline{HP} \leftarrow$ לכל $w \in \Sigma^*$ מתקיים ש:M על x $< |w|$ ולכן $L(M_x)$ היא שפה סופית $\leftarrow \left|L(M_x)\right| < \infty$ ובנוסף M_{STAM} מקבלת כל קלט $\leftarrow \left|L(M_x) \cap L(M_{STAM})\right| < \infty \leftarrow L \notin < M_x, M_{STAM} >$.
- <u>סיכום</u>: ממשפט הרדוקציה $\overline{HP} \notin RE$ ולכן $L \notin RE$. מאחר ו-$R \subseteq RE$ אז גם $L \notin R$. כלומר הוכחנו כי $L \notin RE, R$ כנדרש.

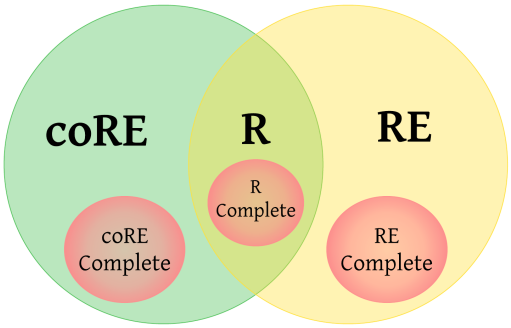
∎

שפות שלמות

הגדרות

<u>הגדרה 1</u> - שפה L היא שלמה ב-RE אם"ם היא מקיימת:

1. $L \in RE$
2. $\forall L' \in RE : L' \leq L$ (L' ניתנת לרדוקציה על L).



<u>הגדרה 2</u> - שפה L היא שלמה ב-R אם"ם היא מקיימת:

1. $L \in R$
2. $\forall L' \in R : L' \leq L$ (L' ניתנת לרדוקציה על L).

<u>הגדרה 3</u> - שפה L היא שלמה ב-$coRE$ אם"ם היא מקיימת:

1. $L \in coRE$
2. $\forall L' \in coRE : L' \leq L$ (כלומר, L' ניתנת לרדוקציה על L).

<u>סימונים</u>

- נסמן $RE - complete$ כאשר $L \in RE - complete$ היא מחלקת השפות השלמות ב-RE.
- נסמן $R - complete$ כאשר $L \in R - complete$ היא מחלקת השפות השלמות ב-R.
- נסמן $coRE - complete$ כאשר $L \in coRE - complete$ היא מחלקת השפות השלמות ב-$coRE$

תרגיל 6

דוגמא לשפה שלמה ב-RE.

<u>טענה</u>: השפה $L_u = \{< M, x > \mid M \ accepts \ x\}$ היא שפה שלמה ב-RE.

<u>מבוא להוכחה</u>: לפי ההגדרה, צריך להראות ש: **(1)** $L_u \in RE$ וגם **(2)** שלא משנה איזה שפה אחרת $L' \in RE$, יש לה רדוקציה ל-L_u.

<u>הוכחה</u>:

1. הוכחנו בשיעורים קודמים ש:$L_u \in RE$ (נחזור על זה שוב):

נתאר מכונת טיורינג M_u המקבלת את השפה L_u:

M_u על קלט $< M, x >$ תסמלץ את ריצת M על הקלט x ותענה כמוה.

- (ראינו שניתן "לסמלץ" ריצה של מכונה שנתונה לנו כקידוד).

נשאר להוכיח את נכונות הבניה:

- אם $< M, x > \in L_u$ אז M מקבלת את הקלט x, מכאן ש-M_u מקבלת את הקלט $< M, x >$ אזי $< M, x > \in L(M_u)$.
- אם $< M, x > \notin L_u$ אז M לא מקבלת את הקלט x, מכאן ש-M_u לא מקבלת את הקלט $< M, x >$ אזי $< M, x > \notin L(M_u)$. (כלומר היא דוחה או לא עוצרת כהגדרה של RE).

הראנו ש: $L_u = L(M_u)$ כלומר שקיימת מכונה M_u <u>המקבלת</u> את השפה L_u ולכן $L_u \in RE$.

2. נראה רדוקציה **מכל שפה** ב-RE אל $L_u = \{< M, x > \mid x \in L(M)\}$.

תהי שפה $L' \in RE$, אזי קיימת עבורה מ"ט M' <u>המקבלת</u> את L'. יהי $< M' >$ קידוד של המכונה M'.
אזי פונקציית הרדוקציה עבור $L' \leq L_u$ היא: $f(x) =< M', x >$.
כאשר M_u על קלט w אז:

1. נריץ את M' על x:
   a. אם M' מקבל את x אז <u>קבל</u>.
   b. אם M' דוחה את x אז <u>דוחה</u>.
   c. אם M' לא עוצר על x אז גם <u>לא עוצר</u>.

נכונות הפונקציה:
הפונקציה מלאה וניתנת לחישוב (שרשור של מחרוזת קבועה למילה הנתונה).
<u>תקפות הרדוקציה</u>:

1. אם $x \in L'$ ( ) M' ( ) מקבלת את x ( ) M_u ( ) עוצר ומקבל את w ( ) $w \in L_u$.
2. אם $x \notin L'$ ( ) M' ( ) דוחה את x או לא עוצרת ( ) M_u ( ) דוחה את w או לא עוצר ( ) $w \notin L_u$.

■

האם יש עוד שפות שלמות ב-RE?

נתונה שפה שלמה $L \in RE - Complete$ ונתונה שפה נוספת $L^* \in RE$.
אם מתקיים ש: $L \leq L^*$ אז נסיק שגם $L^* \in RE - Complete$.
<u>שאלה</u> - למה זה נכון?
<u>תשובה</u> - הראנו שתכונת הטרנזטיביות תקפה עבור רדוקציות. אם לכל $L' \in RE$ אפשר לעשות עם L רדוקציה כלומר $L' \leq L$ וגם מתקיים ש: $L \leq L^*$ אז $L \leq L^* \leq L'$ ולפי טרנזיטיביות $L' \leq L^*$.

<u>מסקנה</u> - השפה HP היא גם שלמה ב-RE.

<u>הוכחה</u> - $L_u \leq HP$. פונקציית הרדוקציה: $f(< M, x >) =< M', x >$.

כך ש: M' זהה למכונה M, רק שאם M דוחה, אז M' נכנסת ללולאה אינסופית.

M' על הקלט w:

- מריצה את M על x:
  - אם M קיבל את x - <u>תקבל</u>
  - אם M דחתה את x - <u>תרוץ לאינסוף</u>.

הוכחת נכונות פונקציית הרדוקציה:

- הפונקציה מלאה וניתנת לחישוב (שרשור של מחרוזת קבועה למילה הנתונה).
- הפונקציה פתירה:
  - אם $< M, x >\in L_u$ מקבלת את $M \leftarrow x$ מקבלת את $M' \leftarrow w$ מקבלת את $< M', x >\in HP$.
  - אם $< M, x >\notin L_u$ דוחה את $M \leftarrow x$ לא עוצרת $M' \leftarrow < M', x >\notin HP$.

לכן, גם השפה HP היא שפה שלמה ב-RE.

שיטה להוכחת שלמות של שפה

נתונה שפה שלמה $L \in RE - Complete$ ונתונה שפה נוספת $L^* \in RE$.

אם מתקיים ש: $L \leq L^*$ אז נסיק שגם $L^* \in RE - Complete$.

<u>תזכורת</u>: ראינו רדוקציה $HP \leq L_\infty$.

<u>שאלה</u>: האם זה אומר ש: L_∞ גם היא שלמה ב-RE?.

<u>תשובה</u>: לא, כי $L_\infty \notin RE$.

תרגיל 7

<u>שאלה</u>: האם יש שפות שלמות ב-R?

<u>תשובה</u>: כן, כמעט כולן! יש בדיוק 2 שפות **שלא** יכולות להיות שלמות ב-R (הם השפות $\emptyset, \Sigma^*$).

<u>שאלה</u>: למה השפות $\emptyset, \Sigma^*$ לא שלמות?

<u>תשובה</u>: כי אי אפשר לעשות עליהן רדוקציה, כי $\emptyset, \Sigma^*$ שתיהן אומרות "כולם בפנים או כולם בחוץ" אז אין לנו אף שפה אחרת שאפשר לעשות עליהן רדוקציה (מלבד אחת על השניה).

טענה

$R - Complete = R \setminus \{\Sigma^*, \emptyset\}$. כלומר כל שפה $L \in R$ היא שלמה מלבד השפות $\{\Sigma^*, \emptyset\}$.

<u>הוכחה</u>: ראשית, נשים לב שלא יכולה להיות קיימת רדוקציה משפה כלשהי L שאינה טריוויאלית אל אחת מהשפות הטריוויאליות $\Sigma^*, \emptyset$.

נוכיח שבהינתן שתי שפות כלשהן: $L_1, L_2 \in R \setminus \{\Sigma^*, \emptyset\}$ מתקיים ש: $L_1 \leq L_2$.

תהי M_1 מכונת טיורינג המכריעה את השפה L_1.

נבחר שתי מילים $x \in L_2, y \notin L_2$ (האם אפשר לבחור? - כן כי אנחנו לא ב-$\Sigma^*, \emptyset$).

נתאר מכונה לחישוב פונקציית הרדוקציה $L_1 \leq L_2$:

המכונה M_f על קלט w:

- מסמלצת את ריצת M_1 על הקלט w:
  - אם M_1 קיבלה את w - מחזירה את x.
  - אם M_1 דחתה את w - מחזירה את y.

<u>הערה</u>: בשלב הזה, פונקציית הרדוקציה החזירה x אם M_1 קיבלה את w. אמרנו שפלט הפונקציה נשלח למכונה של L_2, לכן המכונה M_2 תקבל את x ומכיוון שהגדרנו כבר מראש ש: $x \in L_2$ אז M_2 תקבל את x כנדרש. עבור y השלב הזה דומה רק הפוך (דחייה).

נוכיח את נכונות פונקציית הרדוקציה:

1. הפונקציה מלאה וניתנת לחישוב - תיארנו את האלגוריתם של המכונה המחשבת את הפונקציה. בנוסף, M_1 מכריעה את L_1, כך שלכל מילה M_1 עוצרת. לכן, לכל מילה מ-Σ^*, המכונה M_f תחזיר פלט.
2. הפונקציה תקפה:
   a. אם $x \in L_2$ קיבלת את w $\leftarrow$ M_f מחזירה את x $\leftarrow$ M_1 $x \in L(M_2)$.
   b. אם $y \notin L_2$ דחתה את w $\leftarrow$ M_f מחזירה את y $\leftarrow$ M_1 $y \notin L(M_2)$.

משפט רייס

מבוא

משפט רייס הוא משפט שעוסק ביכולת של אלגוריתמים לחקור אלגוריתמים אחרים. באופן אינטואיטיבי המשפט טוען שתוכנית מחשב אינה יכולה לדעת כמעט כלום על הפלטים של תוכניות מחשב הנתונות לה כקלט.

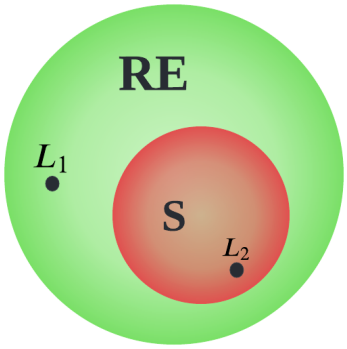
הבעיה פשוטה: בהינתן תכונה של קבוצת מספרים טבעיים, אנחנו רוצים אלגוריתם כלשהו שמקבל כקלט מכונת טיורינג ואומר האם הקבוצה של המספרים שמזוהים על ידי המכונה היא בעלת התכונה הזו או לאו.
כפי שניתן לנחש, אלגוריתם שכזה לא קיים כמעט אף פעם, ולמשפט שאומר זאת בצורה פורמלית קוראים משפט רייס. מה אומר משפט רייס?
שלכל תכונה לא טריוויאלית S של קבוצות טבעיים (ובניסוחים אחרים - פונקציות, שפות - זה לא חשוב כי הכל שקול) לא קיים אלגוריתם שמקבל כקלט מכונת טיורינג M ואומר האם קבוצת המספרים ש-M מקבלת היא בעלת התכונה S.
עד עכשיו, כשראינו שפה מהצורה: $L = \{< M > \mid L(M)\ has\ a\ particular\ property\}$ ניסינו להתאים לה רדוקציה ספציפית כדי להוכיח עבורה לאיזה מחלקה היא איננה שייכת.
במקרים כאלה, משפט רייס נותן לנו קיצור דרך.

הגדרה - תכונה של שפות ב-RE

אבחנה: כשמדברים על תכונה מסוימת, נוח לדבר על הקבוצה של האובייקטים אשר מקיימים את התכונה.
לדוגמה: התכונה "עיניים חומות" ≡ קבוצת האנשים בעלי עיניים חומות.
אנו נתמקד בתכונות של שפות למשל:

1. שפות בעלות לפחות 5 מילים.
2. שפות בעלות מספר אינסופי של מילים.
3. שפות המכילות רק מילים בעלות אורך זוגי.
4. שפות המכילות את המילה ε.

הגדרה - תכונות לא טריוויאליות



משפט רייס עוסק בתכונות של שפות שהמאפיין אותן הוא היותן תתי קבוצות לא ריקות החלקיות ממש ל-RE.
הגדרה: תכונות **לא טריוויאליות** ב-RE הן תכונות S כך ש: $S \subseteq RE$ וכן $\emptyset \neq S \neq RE$.
אבחנה: לכל תכונה לא טריוויאלית S קיימת לפחות שפה אחת ב-S ולפחות שפה אחת ב-$RE \backslash S$.

הגדרה - השפה L_S

בהינתן תכונה $S \subset RE$, נגדיר את השפה $L_S = \{< M > \mid L(M) \in S\}$.

כלומר, L_S היא שפת קידודי המכונות כך ששפת המכונה שייכת לתכונה S.

<u>נסמן להמשך הפרק</u>:

- $\emptyset$ - קבוצת ריקה של שפות.
- φ - השפה הריקה.

<u>דוגמאות</u>:

1. לתכונה $s_1 = \{L \in RE \mid |L| \geq 2\}$ מתאימה השפה $L_{s_1} = \{< M > : |L(M)| \geq 2\}$
2. לתכונה $s_2 = \{L \in RE \mid \varepsilon \in L\}$ מתאימה השפה $L_{s_2} = \{< M > : \varepsilon \in L(M)\}$
3. לתכונה $s_3 = \{\varphi, \Sigma^*, \{11, 0\}\}$ מתאימה השפה $L_{s_3} = \{< M > : L(M) \in s_3\}$
4. לתכונה $s_4 = \{L \in RE : |L| \; is \; infinite\}$ מתאימה השפה $L_\infty = \{< M > : |L(M)| \; is \; infinite\}$

משפט רייס

לכל תכונה לא טריוויאלית $\emptyset \neq S \subset RE$ מתקיים $L_S \notin R$.

<u>במילים</u>: כל תכונה שאפשר לכתוב ב-RE שהיא לא $\emptyset$ והיא לא כל RE אומר ששפת כל המכונות שמקיימות את התכונה S לא שייכת למחלקה R.

משפט רייס המורחב

בנוסף למשפט,

- אם $\varphi \notin S$ (השפה הריקה) אז גם $L_S \notin coRE$.
- אם $\varphi \in S$ (השפה הריקה) אז גם $L_S \notin RE$.

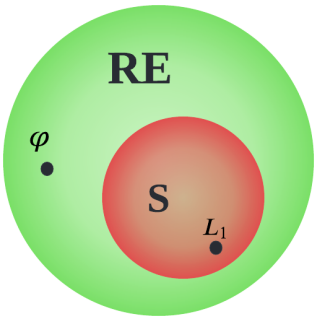
<u>רעיון ההוכחה</u>:

נחלק למקרים:

1. אם $\varphi \notin S$ (השפה הריקה) אז נראה כי $HP \leq L_S$ ומזה נובע כי $L_S \notin coRE$.
2. אם $\varphi \in S$ (השפה הריקה) אז נראה כי $\overline{HP} \leq L_S$ ומזה נובע כי $L_S \notin RE$.

בשני המקרים נקבל כי $L_S \notin R$.

<u>תזכורת</u>: $HP \notin coRE \; and \; \overline{HP} \notin RE$.



<u>הוכחה (מקרה 1)</u>: אם $\varphi \notin S$ נראה כי $HP \leq L_S$ ומזה נוכיח ש: $L_S \notin coRE$.

אם השפה הריקה לא מקיימת את התכונה S (תכונה לא טריוויאלית), כלומר $\varphi \notin S$ אז $\varphi \in RE\backslash S$. לכן, קיימת שפה $L_1 \in S$, ותהי M_{L_1} מ"ט <u>המקבלת</u> את L_1.

(למה L_1 קיימת? - כי ראינו שלכל תכונה לא טריוויאלית S קיימת לפחות שפה אחת ב-S ולפחות שפה אחת ב-$RE\backslash S$). נוכיח ש: $HP \leq L_S$ (נראה רדוקציה מ-HP ל-L_S).

פונקציית הרדוקציה: $f(< M,x >) =< M_x >$.

כך ש: M_x על קלט $y \in \Sigma^*$:

1. מריצה את M על x. (מחסום).
2. מריצה את M_{L_1} על y ועונה כמוה.

<u>הערה</u>: אם M לא עצרה על x אז M_x תקוע לנצח וממשיך לרוץ, לכן השפה תהיה ריקה φ כי אין לנו אף y שאפשר יהיה להריץ אותו על M_{L_1} כי M לא עוצרת. אבל, אם השפה שקיבלנו ריקה φ אז אנחנו לא בתכונה S.

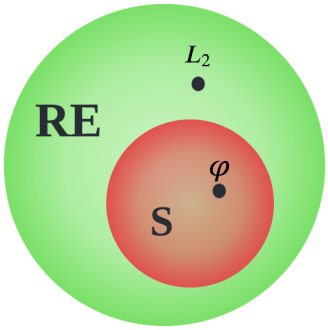
נוכיח את נכונות פונקצית הרדוקציה:

1. הפונקציה מלאה (מהגדרה).
2. ניתנת לחישוב (פעולת קומפילציה פשוטה) כי ניתן לקודד כל מ"ט.
3. תקפה:

- $< M,x > \in HP(\ )\ M\ stops\ on\ x\ (\ )\ M_x\ runs\ M_{L_1}\ on\ y\ (\ )\ L(M_x) = L(M_{L_1})(\ )$
  $(\ )We\ know\ that\ M_{L_1} \in L_S\ (\ )\ M_x \in L_S$
- $< M,x > \notin HP(\ )M\ runs\ forever\ on\ x\ (\ )\ M_x\ runs\ forever\ on\ y\ (\ )$
  $(\ )L(M_x) = \varphi\ (\ )\ We\ know\ that\ \varphi \notin L_S\ (\ )\ M_x \notin L_S$

אנו יודעים כי $HP \notin coRE$ ומאחר והוכחנו ברדוקציה $HP \leq L_S$ אז קיבלנו גם ש: $L_S \notin coRE$ כנדרש.

∎



<u>הוכחה (מקרה 2)</u>: אם $\varphi \in S$ נראה כי $\overline{HP} \leq L_S$ ומזה נוכיח ש: $L_S \notin RE$.

השפה הריקה מקיימת את התכונה S (תכונה לא טריוויאלית) כלומר $\varphi \in S$.

תהי שפה $L_2 \in RE\backslash S$ ותהי M_{L_2} מ"ט <u>המקבלת</u> את L_2.

נוכיח ש: $\overline{HP} \leq L_S$.

פונקציית הרדוקציה: $f(< M,x >) =< M_x >$.

כך ש: M_x על קלט $y \in \Sigma^*$:

1. מריצה את M על x. (מחסום).

2. מריצה את M_{L_2} על y ועונה כמוה.

הערה: אם $\langle M,x\rangle \in \overline{HP}$ אז M לא עצרה על x כלומר M_x תקוע לנצח וממשיך לרוץ, לכן השפה תהיה ריקה $L(M_x) = \varphi$ ולכן היא ב-S.

נוכיח את נכונות פונקצית הרדוקציה:

1. הפונקציה מלאה (מהגדרה).
2. ניתנת לחישוב (פעולת קומפילציה פשוטה) כי ניתן לקודד כל מ"ט.
3. תקפה:

- $\langle M,x\rangle \in \overline{HP}\ (\ )\ M\ runs\ forever\ on\ x\ (\ )\ M_x\ runs\ forever\ on\ y\ (\ )$
  $(\ )\ L(M_x) = \varphi\ (\ )\ We\ know\ that\ \varphi \in S\ (\ )\ M_x \in L_S$
- $\langle M,x\rangle \notin \overline{HP}\ (\ )\ M\ stops\ on\ x\ (\ )\ M_x\ runs\ M_{L_2}\ on\ y\ (\ )$
  $(\ )\ L(M_x) = L(M_{L_2})\ (\ )\ We\ know\ that\ M_{L_2} \notin L_S\ (\ )\ M_x \notin L_S$

אנו יודעים כי $\overline{HP} \notin RE$ ומאחר והוכחנו ברדוקציה $\overline{HP} \leq L_S$ אז קיבלנו גם ש: $L_S \notin RE$ כנדרש.

∎

מה קורה כאשר התכונה S היא טריוויאלית?

אם נתון שתכונה S טריוויאלית אז:

1. אם $S = RE$, אז אפשר לקבל כל קידוד של מ"ט. מכיוון שהנחנו כי כל מחרוזת מתארת מ"ט כלשהי, נקבל כי השפה L_S מכילה את כל המחרוזות, כלומר $L_S = \Sigma^*$ ואכן $\Sigma^* \in R$.
2. אם $S = \emptyset$, אז אפשר לדחות כל קידוד של מ"ט. כלומר, השפה L_S לא מכילה מילים כלל, כלומר $L_S = \varphi$ ואכן $\varphi \in R$.

<u>מסקנה 1</u>: אם התכונה S היא טריוויאלית - משפט רייס "לא עובד".

<u>מסקנה 2</u>: $L_S \in R$ כאשר S טריוויאלית.

מתי מותר להשתמש במשפט רייס?

מותר להשתמש במשפט רייס רק כשהשפה הנתונה לנו L היא אכן קבוצת אובייקטים של תכונה מסוימת של שפות, כלומר, קיימת תכונה S כך ש- $L = L_S = \{\langle M\rangle \mid L(M) \in S\}$.

לפעמים התכונה נראית מסובכת, אבל היא טריוויאלית.

- אם נרצה להגדיר את S כתכונה של מכונה - משפט רייס לא יעבוד.
- אם נרצה להגדיר את S כתכונה של שפה של מכונה - משפט רייס כן יעבוד.

<u>שאלה</u>: איך מוכיחים שתכונה היא <u>לא טריוויאלית</u>?

<u>תשובה</u>: מראים דוגמה לשפה שמקיימת את התכונה ושפה אחרת שלא מקיימת את התכונה.

דוגמה לתכונה של מכונה: $L = \{< M > \mid M\ halts\ on\ every\ input\}$

דוגמה לתכונה של שפה של מכונה: $L = \{w \in \Sigma^*: |w|\ is\ even\}$

אבחנה: אם התכונה המדוברת היא תכונה של מכונה ולא תכונה של שפה של המכונה אז משפט רייס לא עובד.

דוגמה לאבחנה: נתונה השפה $L = \{< M > \mid M\ halts\ on\ every\ input\}$.

עצירה היא תכונה של מכונה ולא של שפה (כי שואלים מתי M עוצרת), בהינתן תכונה לא טריוויאלית S:

- יש מכונות שתמיד עוצרות שהשפה שלהן ב-S.
- יש מכונות שמקבלות את אותה שפה שלא עוצרות על מילים שלא בשפה.

התכונה היחידה של משפט רייס שעשויה להתאים לשפה הנתונה L היא $S = R$.

אבל, אפשר לחשוב על מכונה שמקבלת שפה ב-R ולא עוצרת על מילים שלא בשפה.

שאלה: למה זה לא סותר את זה שהשפה ב-R?

תשובה: כי כדי להוכיח ששפה מסוימת היא ב-R, מספיקה מכונה אחת, אבל יכול להיות שיש מכונה אחרת שמקבלת את אותה השפה אבל היא לא מקיימת את התכונה שהיא תמיד עוצרת. למשל, מכונה שעל כל מילה שלא בשפה - תיכנס ללולאה אינסופית.

במקרה כזה, צריך להוכיח בדרכים שונות (כלומר ע"י רדוקציה).

כמה דוגמאות

דוגמה 1 - נתונה השפה: $L_1 = \{< M > : |L(M)| = 3\}$.

- האם שייכת ל-R ? - **לא**, כי כל מה שעובד בריייס - לא נמצא ב-R.
- האם שייכת ל-RE? - **לא**, צריך להוכיח ברדוקציה - כי משפט רייס לא עוזר פה.
- האם ניתן להשתמש בריייס? - **כן**, כי זה תנאי על השפה. וזה לא טריוויאלי כי אפשר לחשוב על שפות שהגודל שלהם הוא בדיוק 3 ואפשר לחשוב על אחרות שיש להם גודל שונה מ-3.

דוגמה 2 - נתונה השפה: $L_2 = \{< M > : L(M) \subseteq \Sigma^*\}$

- האם שייכת ל-R ? - **כן**, כי זה תכונה טריוויאלית.
- האם שייכת ל-RE? - **כן**, כי כל מה שמוכל ב-R מוכל גם ב-RE.
- האם ניתן להשתמש בריייס? - **לא**, כי זו תכונה טריוויאלית - כל שפה מוכלת ב-Σ^* אז כל מכונה שנקבל השפה של המכונה תיהיה מוכלת ב-Σ^*.

דוגמה 3 - נתונה השפה: $L_3 = \{< M > : M\ accepts\ the\ word\ \varepsilon\}$

- האם שייכת ל-R ? - **לא**, כי זו תכונה לא טריוויאלית.
- האם שייכת ל-RE? - **כן**, אפשר להריץ את M על ε ואז יכול להיות שיתקבל או שלא יעצור.
- האם ניתן להשתמש בריייס? - **כן**, כי זה תנאי על השפה. וזה לא טריוויאלי כי אפשר לחשוב על שפות ש $\varepsilon \in L$ ואפשר לחשוב על אחרות שעבורם $\varepsilon \notin L$.

דוגמה 4 - נתונה השפה: $L_4 = \{< M > : M\ rejects\ the\ word\ \varepsilon\}$

- האם שייכת ל-R ? - **לבדוק**
- האם שייכת ל-RE? - **לבדוק**
- האם ניתן להשתמש ברייס? - **לא**, כי זה תכונה של המכונה. למה? - בשפה L_3 כן אפשר להשתמש ברייס כי אם המילה ε התקבלה אז נעצור. אבל ב-L_4 אם המילה ε נדחתה, אז ייתכן שלא נעצור ונכנס ללולאה אינסופית (לפי הגדרת RE) ולכן זה תלוי במכונה - כי היא הכניסה אותנו לריצה אינסופית.

דוגמה 5- נתונה השפה: $L_5 = \{< M > : \varepsilon \in L(M)\}$

- האם שייכת ל-R ? - **לא**, כי זה לא טריוויאלי.
- האם שייכת ל-RE? - **כן**, אפשר להריץ את M על ε ואז יכול להיות שיתקבל או שלא יעצור.
- האם ניתן להשתמש ברייס? - **כן**, כי זה תכונה של שפה וזה לא טריוויאלי כי אפשר לחשוב על שפות L ומצד שני על שפות אחרות כך ש: $\varepsilon \notin L$ ו $\varepsilon \in L$.

דוגמה 6 - נתונה השפה: $L_6 = \{< M > : L(M) \in coRE\}$

- האם שייכת ל-R ? - **לא**, כי אם זה לא ב-RE זה בוודאות לא ב-R.
- האם שייכת ל-RE? - **לא**, כי לפי רייס מורחב אם $\varphi \in S$ אז למדנו שבמקרה זה $S \notin RE$. ידוע כי השפה הריקה נמצאת ב-R ולכן היא בהכרח גם ב-$coRE$ כי היא מוכלת בתוכה.
- האם ניתן להשתמש ברייס? - **כן**, כי זה תכונה של שפות והיא תכונה לא טריוויאלית כי $HP \notin coRE$ ויש לנו שפה אחרת $\varphi \in coRE$ ולכן התכונה לא טריוויאלית.

דוגמה 7 - נתונה השפה: $L_7 = \{< M > : L(M) \notin R\}$

- האם שייכת ל-R ? - **לא**, כי זה לא שייך גם ל-RE.
- האם שייכת ל-RE? - **לא**, כי $\varphi \notin L_7$. אפשר להוכיח ברדוקציה על $\overline{HP}$ כי למדנו כבר ש $\overline{HP} \notin RE$.
- האם ניתן להשתמש ברייס? - **כן**, כי זה תכונה של שפות והיא תכונה לא טריוויאלית כי אפשר לבחור ב-$HP \notin R$ ובשפה אחרת $\varphi \in R$.

דוגמה 8 - נתונה השפה: $L_8 = \{< M > : |L(M)|\ is\ odd\ and\ |< M >| \leq 1000\}$

- האם שייכת ל-R ? - **כן**, כי כל שפה סופית היא רגולרית וכל הרגולריות מוכלות ב-R.
- האם שייכת ל-RE? - **כן**, כי זה שייך גם ל-R.
- האם ניתן להשתמש ברייס? - **לא**, כי זה טריוויאלי - השפה סופית שייכת ל-R.

תרגול 3 - שאלה 5

נתונה השפה $L_3 = \{< M > : |L(M)| \geq 3\}$. כלומר, המכונה מקבלת לפחות 3 מילים.

<u>אינטואיציה</u>: נרצה להבין מה התכונה של השפה ולפיה להבין לאיזה מחלקה השפה הזו שייכת. ייתכן שהמכונה תקבל פחות מ-3 מילים אבל היא תמשיך לבדוק את שאר המילים, לא תקבל אף אחת מהם ותרוץ לאינסוף. כלומר, לא ניתן להכריע את תנאי השפה ולכן ננית כי $L_3 \notin R$ אך אכן ב-RE. אנחנו צריכים להוכיח שאכן $L_3 \notin R$ וגם ש: $L_3 \in RE$.

<u>הוכחה 1</u>: נוכיח כי $L_3 \in RE$ בעזרת בניית מ"ט המקבלת את השפה L_3. (להשלים בבית).

<u>הוכחה 2</u>: נוכיח כי $L_3 \notin R$ באמצעות משפט רייס (זה הרבה יותר קל מרדוקציה).

1. <u>הגדרת תכונה S</u>: התכונה היא $S = \{L \in RE: |L| \geq 3\}$ ואכן $L_S = L_3$.
2. <u>נוכיח ש-S לא טריוויאלית</u>: כלומר, יש להראות דוגמה ללפחות שפה אחת $L_1 \in S$ המקיימת את התכונה, ושפה אחרת $L_2 \notin S$ שלא מקיימת אותה. כאשר $L_1, L_2 \in RE$.

   <u>הערה</u>: ננסה לרוב להדגים עם $\emptyset$ וגם Σ^* כי הכי קל לנו לעבוד איתם (לרוב זה יצליח) וגם כי $\emptyset, \Sigma^* \in RE$.

   נשים לב ש: $\emptyset \notin S$ (כי הגודל של השפה הריקה הוא 0 ולכן לא מקיים את התכונה) וגם $\Sigma^* \in S$.
   לכן S לא טריוויאלית.
3. <u>מסקנה</u>: לפי משפט רייס המורחב נובע ש אם $\emptyset \notin S$ אז גם $L_S \notin coRE$ ובפרט $L_S \notin R$.
   בנוסף, הראנו שמתקיים ש: $L_3 = L_S$ ולכן גם $L_3 \notin R$ כנדרש.

תרגול 3 - שאלה 6

נתונה השפה $L = \{< M > : |L(M)| = 3\}$. כלומר, המכונה מקבלת 3 מילים בלבד.

<u>אינטואיציה</u>:

- דבר ראשון, לא ניתן להכריע את השפה L מכיוון שאי אפשר <u>להכריע</u> מתי נקבל רק 3 מילים בשפת המכונה, כי המכונה עלולה לא לעצור.
- דבר שני, גם אם המכונה M תקבל את המילה שלישית שלה בקלט ה-i, עדיין ייתכן שהיא תקבל עוד מילה בקלט ה-$i+1$. במילים אחרות, המכונה צריכה לעבור על כל Σ^* שזה אינסוף מילים בשביל לקבוע בוודאות שהמכונה קיבלה רק 3 מילים, לכן זו שפה שלא קיים לה מכונה <u>שתקבל</u> אותה.
- לכן, האינטואיציה היא $L \notin R$ וגם $L \notin RE$.

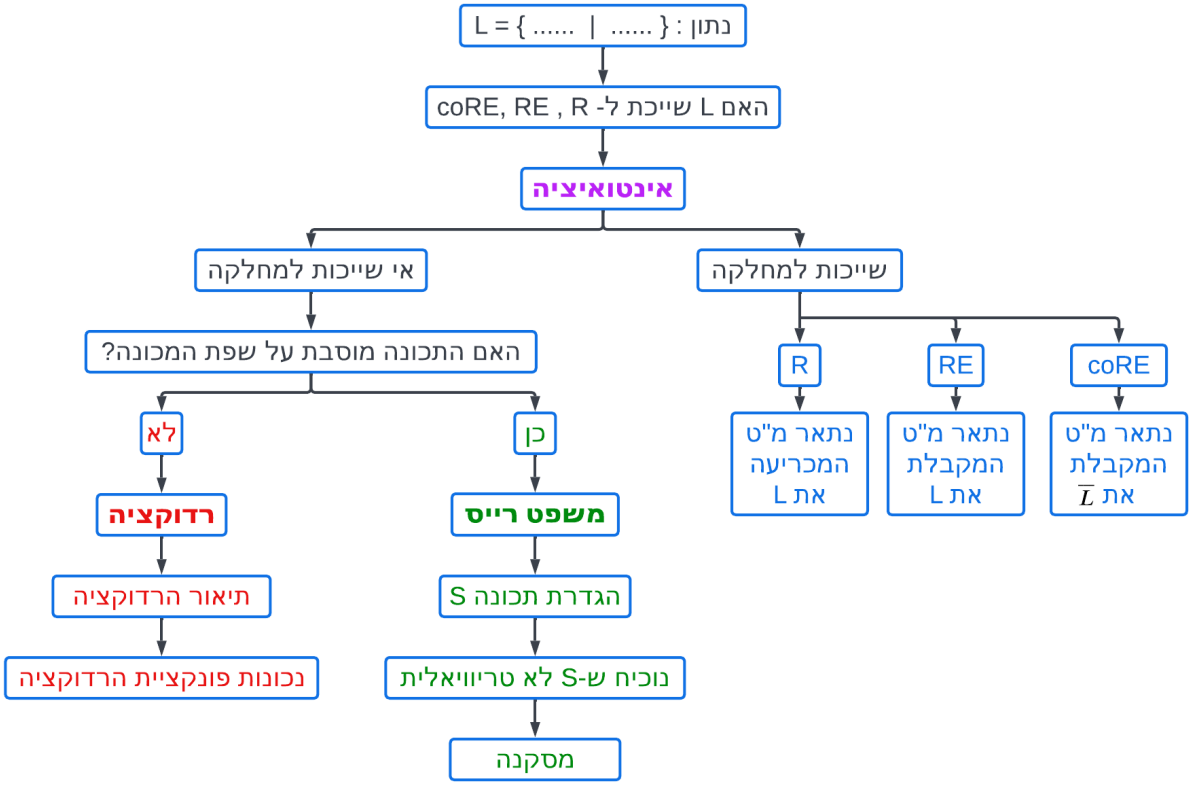
<u>הוכחה 1</u>: נוכיח כי $L \notin R$ באמצעות משפט רייס (זה הרבה יותר קל מרדוקציה).

1. <u>הגדרת תכונה S</u>: התכונה היא $S = \{L': |L'| = 3\}$ ואכן $L_S = L$.
2. <u>נוכיח ש-S לא טריוויאלית</u>: נשים לב ש: $\emptyset \notin S$ (כי הגודל של השפה הריקה הוא 0 ולכן לא מקיים את התכונה) וגם $L' = \{0, 1, 01\} \in S$. לכן S לא טריוויאלית.
3. <u>מסקנה</u>: $L \notin R$. משל.

<u>הוכחה 2</u>: נוכיח כי $L \notin RE$.

1. <u>אינטואיציה</u>: התכונה והאי-טריוויאליות זהה להוכחה 1. אך נשים לב ש: $\emptyset \notin S$ ולפי משפט רייס המורחב $L \notin coRE$, אך זה לא אומר ש: $L \in RE$. לכן, משפט רייס לא מספיק כדי להוכיח שאכן $L \notin RE$. השיטה השניה עבור הוכחת אי-שייכות היא באמצעות רדוקציה.
כדי לעבוד עם רדוקציה, צריך לחשוב על שפה שהוכחנו כבר במהלך הקורס שלא שייכת ל-RE.
השפה הזו היא השפה $\overline{HP} = \{< M, x > \ : \ M \ doesn't \ stop \ on \ x\}$.
הרעיון הוא כזה:
   - אם M לא עוצר על x אז M_x צריכה לקבל 3 מילים בלבד וכך $< M_x > \in L$.
   - אם M עוצר על x אז M_x צריכה לקבל אינסוף מילים וכך $< M_x > \notin L$.
2. <u>תיאור הרדוקציה</u>: נראה רדוקציה $\overline{HP} \leq L$.
   a. פונקציית הרדוקציה מוגדרת כך: $f(< M, x >) =< M_x >$.
   b. M_x על w פועלת כך:
      i. M_x מקבלת כל $w \in \{0, 1, \varepsilon\}$.
      ii. תסמלץ את M על x.
      iii. אם M עצרה אז M_x תקבל את w.
3. <u>הוכחת פונקציית הרדוקציה</u>:
   a. פונקציה מלאה: לכל M, x קיים מכונה M_x.
   b. פונקציה ניתנת לחישוב: לכל מ"ט קיים קידוד בצורת מחרוזת.
   c. פונקציה פתירה:
      i. אם $< M, x > \in \overline{HP}$ לא עוצרת על $x \leftarrow M_x$ מקבלת רק $w \in \{0, 1, \varepsilon\} \leftarrow L(M_x) = \{0, 1, \varepsilon\} \leftarrow \left|L(M_x)\right| = 3 \leftarrow L \in < M_x >$ כנדרש.
      ii. אם $< M, x > \notin \overline{HP}$ עוצרת על $x \leftarrow M_x$ מקבלת כל $w \in \Sigma^* \leftarrow L(M_x) = \Sigma^* \leftarrow$ קיבלנו ששפת המכונה בגודל אינסופי $\leftarrow L \notin < M_x >$.
4. <u>סיכום</u>: הוכחנו ש: $\overline{HP} \notin RE$, בהפעלת רדוקציה $\overline{HP}$ על L אז גם $L \notin RE$.

סכמת עבודה כללית



תרגול 3 - שאלה 7

נתונה השפה: $L = \{< M > : L(M) \in RE\}$.

האם $L \in R$, האם $L \in coRE$, האם $L \in RE$?

<u>פתרון</u>: $L \in R$, אבל לא בגלל רייס! למעשה, $L = \Sigma^*$ מההגדרה של RE.

תרגול 3 - שאלה 8

נתונה השפה: $L = \{(< M_1 >, < M_2 >, x) \mid M_1 \text{ stops on } x \text{ and } M_2 \text{ doesn't stops on } x\}$

לאיזו מחלקה השפה שייכת?

<u>פתרון</u>:

<u>אינטואיציה</u>:

- אי אפשר להשתמש במשפט רייס מכיוון שהשפה תלויה במכונה ולא בשפה שלה.
- זה לא ב-R כי אי אפשר לדעת מתי לעצור לכל x כזה. בנוסף אפשר לראות כי יש פה שילוב של שפות שכבר הוכחנו שהם HP, $\overline{HP}$ שדומות ל-L וכבר הוכחנו עליהם שלא שייכות ל-R.
- יש לנו חיבור של שפה $HP \in RE$ עם שפה $\overline{HP} \in coRE$ וחיבור כזה יכול להוביל אותנו שגם אם שיכול להיות שנקבל ולא נעצור וגם נדחה ולא נעצור, כלומר ב-2 המקרים ייתכן שלא נעצור.
- האינטואיציה אומרת לנו שייתכן שמדובר בשייכות למחלקה $\overline{RE \cup coRE}$.

- נוכיח על ידי שתי רדוקציות $HP \leq L$ וגם $\overline{HP} \leq L$.

<u>הוכחת רדוקציה ראשונה</u>: $HP \leq L$.

- <u>אינטואיציה</u>: אנחנו צריכים מכונה שתעצור על x ועוד מכונה שלא תעצור על x. המכונה שתעצור על x מבוססת על מכונות שמקבלות את השפה HP בעוד שהמכונות שלא עוצרות לכל x נסמן ב-M_{loop}.
  אם המכונה M מקבלת את HP אז מעולה היא אכן תעצור על x, אין לנו צורך לתאר אותה.
  כן נדאג לתאר את המכונה M_{loop} שתפקידה להיכנס ללולאה אינסופית לכל קלט w.
- <u>תיאור הרדוקציה</u>:
  - פונקציית הרדוקציה: $f(< M, x >) = \left(< M >, < M_{loop} >, x\right)$
  - הפעלה: M_{loop} על קלט w תפעל כך:
    - נכנסת ללולאה אינסופית
- <u>הוכחת נכונות הפונקציה</u>:
  - הפונקציה מלאה: כי צריך רק להעתיק את $M,\ M_{loop}, x$
  - הפונקציה ניתנת לחישוב: לכל מ"ט יש קידוד.
  - הפונקציה פתירה:
    - $M \leftarrow < M, x > \in HP$ עוצרת על x ו-M_{loop} לא עוצרת $\leftarrow$ $\left(< M >, < M_{loop} >, x\right) \in L$.
    - $M \leftarrow < M, x > \notin HP$ לא עוצרת על x ו-M_{loop} לא עוצרת $\leftarrow$ $\left(< M >, < M_{loop} >, x\right) \notin L$.
- <u>סיכום</u>: הראינו רדוקציה HP על L וכיוון ש: $HP \notin coRE$ אז גם $L \notin coRE$ כנדרש.

<u>הוכחת רדוקציה שניה</u>: $\overline{HP} \leq L$.

- <u>אינטואיציה</u>: נראה רדוקציה $\overline{HP}$ על L ונוכיח כי אם $\overline{HP} \notin RE$ אזי גם $L \notin RE$.
  אנחנו צריכים מכונה שתעצור על x ועוד מכונה שלא תעצור על x.
  מכונה שתמיד עוצרת תהיה M_{STAM} המקבלת כל דבר.
- <u>תיאור הרדוקציה</u>:
  - פונקצית הרדוקציה: $f(< M, x >) = \left(< M_{STAM} >, < M >, x\right)$
  - הפעלה: מדובר במכונות מוכרות שהוכחנו כבר בעבר.
- <u>הוכחת נכונות הרדוקציה</u>:
  - הפונקציה מלאה: צריך רק להעתיק $< M_{STAM} >, < M >, x$ היא מחרוזת קבועה.
  - הפונקציה ניתנת לחישוב: אפשר לקודד כל מכונה.
  - הפונקציה פתירה:

- $< M, x > \in \overline{HP}$ לא עוצרת על x ו-M_{STAM} עוצרת תמיד $\leftarrow M \leftarrow < M, x >$
  $\left(< M_{STAM} >, < M >, x\right) \in L$.
- $< M, x > \notin \overline{HP}$ עוצרת על x ו-M_{STAM} עוצרת תמיד $\leftarrow M \leftarrow < M, x >$
  $\left(< M_{STAM} >, < M >, x\right) \notin L$.

- <u>סיכום</u>: הראינו רדוקציה $\overline{HP}$ על L וכיוון ש: $\overline{HP} \notin RE$ אז גם $L \notin RE$ כנדרש.

לכן, לפי משפט הרדוקציה נקבל ש: $L \in \overline{RE \cup coRE}$ כנדרש.

∎

בעיית העצירה החסומה - BHP

מבוא

בפרק זה נתעסק בשפות עצירה כאשר העצירה יכולה להתבצע מסיבות שונות, כמו למשל:

1. **עצירה תחת הגבלת מספר צעדים** - כמו למשל השפה L_c שנציג בפרק. (צעדי המכונה/האלגוריתם).
2. **עצירה תחת הגבלת מקום** - כמו למשל השפה BHP שנציג בפרק. (הגבלת המקום קשורה לסרט).

השפה L_c

נתונה השפה $L_c = \{< M, x > \mid M\ halts\ on\ x\ within\ c \in \mathbb{N}\ steps\}$.

במילים: כל הקידודים $< M, x >$ כך ש-M עוצרת על x בתוך c צעדים.

אינטואיציה: למשל L_{1000} זה כל הקידודים $< M, x >$ כך שבצעד ה-1000 המכונה M עוצרת על x.

השפה $L_c \in R$ בגלל שאם נתון לנו ה-c אז יש לנו מעין "חסם", ולכן אפשר להכריע אותה כי אנחנו "נאחזים" ב-c שתמיד מוגדר וסופי לכל מספר טבעי שנקבל עבורו. שלב 1 בהוכחה תמיד מסתיים, כי גם אם המכונה אמורה לרוץ לאינסוף צעדים, בגלל שקבענו חסם עבור מספר הצעדים שלנו, אז המכונה תמיד תעצור.

השפה L_c שייכת למחלקה RE.

טענה: $L_c \in R$.

הוכחה: נוכיח כי $L_c \in R$.

נתאר מכונה מכריעה:

U על קלט $< M, x >$ תבצע:

1. הרץ את M על הקלט x למשך c צעדים.
2. אם M עצרה על x - קבל.
3. אם M לא עצרה על x - דחה.

נכונות הבניה:

- אם $< M, x > \in L_c$ $\leftarrow$ M עוצרת על x בצעד ה-c $\leftarrow$ U קיבל $\leftarrow$ $< M, x > \in L(U)$.
- אם $< M, x > \notin L_c$ $\leftarrow$ M לא עוצרת על x או M עוצרת אחרי יותר מ-c צעדים $\leftarrow$ U לא תזהה עצירה ותדחה $\leftarrow$ $< M, x > \notin L(U)$.

הוכחנו כי $L_c = L(U)$ וכן U מכריעה את השפה L_c ומכאן $L_c \in R$ כנדרש.

∎

השפה BHP

נתונה השפה:

$$BHP = \{< M, x > \mid M\ halts\ on\ x,\ and\ M\ doesn't\ enter\ the\ (|x|^2 + 1)'th\ cell\ of\ its\ tape\}$$

<u>במילים</u>: תוודא ש-M מסמלצת על x ועוצרת. ובנוסף, M לא תעבור את התא ה-$(|x|^2 + 1)$ בסרט שלה.

<u>טענה</u>: $BHP \in R$.

<u>מבוא להוכחה</u>: כדי להוכיח $BHP \in R$, נשים לב שמותר למכונה להשתמש בלכל היותר $|x|^2$ תאי זיכרון.

- המכונה יכולה לא לעצור. מהם כל האפשרויות עבור המכונה (עוצרת או לא) וכיצד ניתן לזהות אותם?
  1. כאשר M נכנסת לתא האסור (בין אם עוצרת או לא).
  2. כאשר M עוצרת מבלי להיכנס לתא האסור.
  3. כאשר M נכנסת ללולאה אינסופית בתוך התאים המותרים.
- <u>תזכורת</u>: קונפיגורציה היא תיאור המצב הכללי של המכונה בנקודה הספציפית $c = (\alpha,\ q,\ i)$. כאשר α הוא תוכן הסרט, q הוא המצב הנוכחי ו- i הוא המיקום (האינדקס) של הראש של הסרט.

במקרה שלנו, מכיוון שכמות התאים המותרים לשימוש מוגבלת ע"י $|x|^2$, ניתן לחשב חסם עליון למספר הקונפיגורציות השונות האפשריות.

- <u>למה הקונפיגורציות האלה מעניינות אותנו?</u>: אנחנו רוצים לזהות את כל האפשרויות שהמכונה תבצע בשביל שנוכל להכריע את השפה, אמרנו שאחד מהאפשרויות האלה היא כאשר M נכנסת ללולאה אינסופית בתוך התאים המותרים. לכן כדי לזהות את המקרים שנכנס ללולאה אז אפשר להשתמש בקונפיגורציות. בהרצאה 1, הוכחנו עם עקרון שובך היונים שאם עברנו על כל הקונפיגורציות האפשריות, אז הקונפיגורציה הבאה תסגור לנו מעגל כי נפגשנו בקונפיגורציה הזאת בעבר. לכן, נשתמש בשיטה הזאת כדי לזהות כאשר המכונה M נכנסת ללולאה ואז נוכל לעצור.
- <u>חסם לכמות הקונפיגורציות השונות האפשריות בתוך $|x|^2$ תאים</u>: נרצה לחשב את מספר כל הקונפיגורציות האפשריות שלנו:
  - מספר כל הקלטים האפשריים הוא $\#(\alpha) \leq |\Gamma|^{|x|^2}$.
  - מספר כל הקבוצות האפשריות הוא $\#(q) \leq |Q|$.
  - מספר כל התאים האפשריים על הסרט הוא $\#(i) \leq |x|^2$.

לכן החסם הכללי T לכמות הקונפיגורציות השונות: $T = |\Gamma|^{|x|^2} \times |Q| \times |x|^2$.

- <u>אבחנה</u>: אם נריץ את המכונה ל-$T + 1$ צעדים אז בהכרח קיימת קונפיגורציה שהמכונה ביקרה פעמיים.
- אם המכונה הייתה באותה קונפיגורציה פעמיים, אפשר להסיק שהמסלול של המכונה מכיל מעגל. נניח שקיימים i, j כך ש: $c_i = c_j$. לכן, קיום המעגל נובע מכך שפונקציית המעברים היא דטרמיניסטית.

<u>אינטואיציה</u>: הצגנו במבוא להוכחה כי ניתן לזהות כל מקרה אפשרי של המכונה ולכן עבור כל זיהוי שכזה נוכל להבטיח שהאלגוריתם תמיד יעצור וכך המכונה תכריע את השפה. כלומר $BHP \in R$.
אנחנו הצגנו במבוא 3 אפשרויות וכעת נראה איך האלגוריתם בהוכחה נותנת עבורם מענה:

- כאשר M נכנסת לתא האסור (בין אם עוצרת או לא) - אז נדחה בשלב (a) באלגוריתם.
- כאשר M עוצרת על x מבלי להיכנס לתא האסור - אז נקבל בשלב (b) באלגוריתם.
- כאשר M נכנסת ללולאה אינסופית בתוך התאים המותרים - אז נדחה בשלב (3) באלגוריתם.

<u>הוכחה</u>: נוכיח כי $BHP \in R$. נתאר מכונה מכריעה:
U על קלט $< M, x >$ תפעל:

1. תסמלץ את M על הקלט x למשך $T + 1$ צעדים. (בטוח נכנס ללולאה).
2. בכל צעד:
   a. אם M נכנסה לתא ה-$|x|^2 + 1$ (התא האסור) - <u>דחה</u>.
   b. אם M עצרה על x - <u>קבל</u>.
3. אם הסתיימו $T + 1$ צעדים ללא עצירה - <u>דחה</u>.

נוכיח את נכונות הבניה (צריך להוכיח כי $BHP = L(U)$):

- אם $< M, x > \in BHP$ ← M עוצרת על x תוך לכל היותר T צעדים וגם לא עוברת את התא ה-$|x|^2$
  ← המכונה U תקבל ($< M, x > \in L(U)$).
- אם $< M, x > \notin BHP$ ←
  ↲ <u>מקרה א'</u>: M חורגת לתא ה-$|x|^2 + 1$ לפני הצעד ה-$T + 1$.
  ↲ <u>מקרה ב'</u>: M נכנסת ללולאה בתוך ה-$|x|^2$ התאים החוקיים ולכן M יכולה לבצע יותר מ-$T + 1$ צעדים. המכונה U תסיים את ריצת כל ה-$T + 1$ צעדים בלי לזהות עצירה.
  ← בשני המקרים U תדחה ← ($< M, x > \notin L(U)$).

הוכחנו ש: $BHP = L(U)$ ובנוסף הראנו שהמכונה U היא מכונה מכריעה כי היא תמיד עוצרת ולכן $BHP \in R$
∎

שאלה לתרגול

נתונה השפה $L = \{< M > \mid when\ running\ on\ the\ input\ \varepsilon,\ M\ doesn't\ enter\ the\ 10'th\ cell\}$.
הוכיחו כי $L \in R$.

<u>הוכחה</u>: נוכיח כי $L \in R$. החסם הכללי T לכמות הקונפיגורציות השונות: $T = |\Gamma|^{|x|^2} \times |Q| \times |x|^2$.
נתאר מכונה מכריעה:
U על קלט $< M >$ תבצע:

1. תסמלץ את M על ε למשך $T + 1$ צעדים.
2. בכל צעד:
   a. אם M נכנסה לתא 10 - <u>דחה</u>.

b. אם M עצרה - קבל.

3. אם M לא עצרה אחרי $T + 1$ צעדים - קבל.

נוכיח את נכונות הבניה (צריך להוכיח כי $L = L(U)$):

- $\leftarrow < M > \in L$
  - מקרה א': M עצרה בתוך לכל היותר T צעדים ולא עברה את התא ה-9.
  - מקרה ב': M לא עצרה בתוך גם לאחר $T + 1$ צעדים ולא עברה את התא ה-9.

  $\leftarrow$ ב-2 המקרים U מקבלת $\leftarrow < M > \in L(U)$.
- $\leftarrow < M > \notin L$ נכנסה לתא ה-10 $\leftarrow$ המכונה U תדחה $\leftarrow < M > \notin L(U)$.

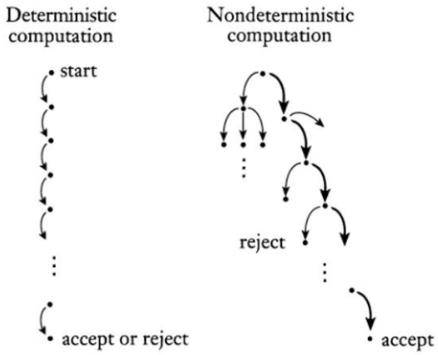
הוכחנו ש: $L = L(U)$ ובנוסף הראנו שהמכונה U היא מכונה מכריעה כי היא תמיד עוצרת ולכן $L \in R$.

∎

<u>הערה לפתרון</u>: התרגיל הזה בוחן את הערנות שלנו. נשים לב שלשפה הנתונה L אין התייחסות בכלל לעצירה, החסם היחידי הוא בנפח, כלומר בהגבלת השימוש של התאים של הסרט. לכן, אנחנו נקבל בין אם M עצרה בין התאים 1-9 או בין אם לא עצרה. למה עדיין היה חשוב לנו להתייחס לעניין ה-$T + 1$? כי אנחנו רוצים להוכיח שקיים מכונה <u>שמכריעה</u> את השפה, כדי שהמכונה תכריעה היא חייבת לזהות כל תרחיש אפשרי עבורה למשל מתי היא עוצרת או לא וכו'...

מכונות טיורינג אי-דטרמיניסטיות

מבוא



- כל אלגוריתם ניתן לתיאור על ידי מודל מתמטי מופשט המכונה **מכונת טיורינג**.
- **<u>מכונת טיורינג הסטנדרטית (דטרמיניסטית)</u>** היא מכונת מצבים מוגדרת היטב, כלומר לכל מצב של המכונה ברור באופן מוחלט מה יהיה הצעד הבא של המכונה.
- **<u>מכונת טיורינג לא-דטרמיניסטית</u>** היא הרחבה של המודל הסטנדרטי, בה הצעד הבא של המכונה הוא מצב מסוים מתוך קבוצה של מצבים אפשריים, הנבחר בצורה "שרירותית".
- מכונה אי-דטרמיניסטית היא בעיקר כלי חשיבתי, שאינו ניתן למימוש בפועל.
- לא ברור האם למכונות אי-דטרמיניסטיות יש "יעילות חישוב" יותר טובה מאשר מכונות דטרמיניסטיות.
- השאלה האם מכונות אי-דטרמיניסטיות מחשבות בעיות "יותר מהר" היא שאלה פתוחה חשובה במדעי המחשב ומכונה **בעיית P=NP**.

<u>דוגמה פרקטית לשימוש במ"ט אי-דטרמיניסטית</u>: נביט לדוגמה על מכונה א"ד למציאת מעגל בגרף.

- בהינתן גרף מכוון, מעגל הוא מסלול לא-ריק אשר מתחיל ומסתיים באותו צומת.
- אלגוריתם N לא-דטרמיניסטי לדוגמה, יקבל כקלט את הגרף G, "ינחש" צומת מוצא q_s, ובכל צעד "ינחש" את הצומת הבא במעגל. אם המכונה חזרה לצומת ממנו התחילה, היא מסיימת במצב מקבל.
- אם קיים מעגל, הרי שמכונה שתנחש צומת שנמצא על המעגל, ובכל צעד תנחש את הצומת הבא במסלול המעגלי, תגיע בסופו של דבר אל צומת המוצא q_s, ותסיים במצב מקבל.
- לכן קלט G כנ"ל מוגדר כקלט שהתקבל על ידי המכונה, כלומר, יש בו מעגל (יש לשים לב, ששאר הניחושים חסרי משמעות, מכיוון שנדרש שיהיה קיים חישוב אחד שיסיים במצב מקבל).
- מצד שני, אם אין בגרף הקלט G מעגל, אף ניחוש לא יכול להחזיר אותנו אל צומת המוצא q_s. כלומר כל החישובים לא יסתיימו במצב מקבל, והגרף מוגדר כקלט שנדחה על ידי המכונה, כלומר שאין בו מעגל.

<u>מכונות לא דטרמיניסטיות</u>

למדנו בקורס "אוטומטים" על מודלים לא דטרמיניסטים:

- אוטומט סופי לא דטרמיניסטי (NFA) **שקול בכוחו** לאוטומט סופי דטרמיניסטי (DFA).
- אוטומט מחסנית לא דטרמיניסטי **חזק יותר** מאוטומט מחסנית דטרמיניסטי.

פונקצית המעברים של מכונת טיורינג אי-דטרמיניסטית

במצב נתון בו מכונה נמצאת במצב q והראש הקורא-כותב שלה קורא את הסמל a, אז היא יכולה לבצע **אחד מכמה צעדים אפשריים**. הפונקציה מוגדרת בצורה הבאה:

$$\delta: Q' \times \Gamma \to P(Q \times \Gamma \times \{L, S, R\})$$

הערות:

- הקבוצה $Q' = Q \backslash F$ כלומר היא קבוצת מצבים לא סופיים.
- קלט: מצב לא סופי $q \in Q'$ ואות $\sigma \in \Gamma$
- פלט: ומחזירה לנו שלשה של מצב $p \in Q$, אות $\beta \in \Gamma$ ותזוזה $d \in \{L, S, R\}$ כלומר (p, β, d). אבל עבור אותו הקלט היא יכולה להחזיר כמה שלשות שונות (לפי ניחוש בבחירה שרירותית).
- לפי ההגדרה הפורמלית, אפשרית גם קבוצה ריקה של צעדים אפשריים, אבל אנחנו נניח שתמיד יש לפחות צעד אפשרי אחד.

שפה של מכונת טיורינג אי-דטרמיניסטית

נאמר שמכונת טיורינג אי-דטרמיניסטית N **מקבלת** מילה x אם בריצה של N על x **קיימת** סדרת צעדים חוקית המסתיימת במצב q_{accept}.

השפה של מ"ט N לא דטרמיניסטית מוגדרת בדומה לשפה של מ"ט דטרמיניסטית:

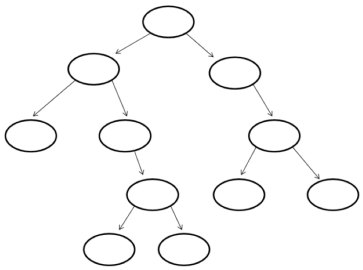
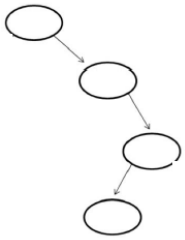
$$L(N) = \{x \in \Sigma^* \mid N \text{ accepts } x\}$$

- במכונה דטרמיניסטית, אמרנו שכל הרצה של המכונה על קלט מסוים היא בעצם סדרה של קונפיגורציות עוקבות, כלומר - **מסלול חישוב**.
- בדומה, במכונה אי-דטרמיניסטית, כל הרצה של מכונה על קלט מסוים נותנת **עץ חישוב**. כאשר ההרצה הספציפית היא בחירה שרירותית של מסלול **יחיד** בעץ החישוב.
- העץ המתקבל במ"ט א"ד הוא עץ בינארי (לא בהכרח מלא).

תזכורת חשובה - הכוח של מודלים לא דטרמיניסטים נובע מן ההגדרה שמילה מתקבלת אם יש לה מסלול חישוב מקבל אחד לפחות (מסלול חישוב שמסתיים במצב המקבל). זה תקף גם אם מסלולי חישוב רבים לא מסתיימים במצב מקבל. (ייתכן אינסוף מסלולי חישוב כאלה).

שאלה לדיון: מתי מילה לא מתקבלת ע"י מכונה לא דטרמיניסטית?
תשובה: אם לא קיים אף מסלול בעץ החישוב שיקבל את המילה.

מסלול חישוב לעומת עץ חישוב

עץ חישוב	מסלול חישוב
	
<u>חישוב של מ"ט לא דטרמיניסטית</u>: אפשר להסתכל על החישוב כעל **עץ** במרחב הקונפיגורציות: 1. מתחילים בקונפיגורציה ההתחלתית. 2. לכל קונפיגורציה יכולות להיות **כמה** קונפיגורציה עוקבות.	**<u>חישוב של מ"ט דטרמיניסטית</u>**: אפשר להסתכל על החישוב כעל **מסלול** במרחב הקונפיגורציות: 1. מתחילים בקונפיגורציה ההתחלתית. 2. לכל קונפיגורציה יש קונפיגורציה עוקבת **אחת** לכל היותר.

אי-סימטריה של מכונות לא דטרמיניסטיות

<u>אבחנה חשובה</u>: אם הרצנו מכונה אי-דטרמיניסטית על קלט x, יש כמה אפשרויות:

1. <u>המכונה קיבלה</u> - (עצרה ב-q_{accept}) ואפשר להסיק ש-x שייך לשפת המכונה.

   אינטואיציה: בביטוי בוליאני $B = (?) \vee (?) \vee (T)$ אז מספיק ונפגשנו ב-T בהתחלה כדי לקבל.
2. <u>המכונה דחתה</u> - ואי אפשר לדעת אם x שייך לשפה או לא כי יכול להיות שקיים מסלול אחר שהמכונה כן יכולה לעצור בו ב-q_{accept} ולקבל. (בהמשך נראה שצריך הרצה מבוקרת למקרים כאלה).

   אינטואיציה: בביטוי בוליאני $B = (?) \vee (?) \vee (F)$ לא מספיק להיפגש ב-F ראשון, צריך הכל F.
3. <u>המכונה נכנסה ללולאה</u> - אי אפשר לדעת כלום...

דוגמה 1

<u>הגדרה</u>: מעגל המילטוני הוא מסלול בגרף העובר בכל צומת פעם אחת פרט לצומת שממנו יצא.

<u>תרגיל</u>: בנו מ"ט א"ד המקבלת כל גרף המכיל מעגל המילטון והוכיחו נכונות.

<u>אינטואיציה</u>: אנחנו רוצים מ"ט שמקבלת קידוד של גרף. אם יהיה בגרף הזה מעגל המילטוני אז נדע לזהות ולקבל ואם אין אז נדחה בכל המסלולים.

- אם בגרף יש מסלול המילטון, אז לכל פרמוטציה מתוך $n!$ אם יש סידור כלשהו שיהווה מעגל המילטוני, אז נכנס למסלול הזה, נראה שכל הצלעות קיימות לפי ההגדרה של המעגל ונעבור למצב מקבל.
- מצד שני, אם אין מסלול המילטוני אז כל פרמוטציה לא תהווה לנו מעגל המילטוני בגרף ולכן נדחה לכל מסלול.
- בתרגילים כאלה, כשנרצה קיימות של תכונה בגרף, אז ננחש סידור כלשהו של הגרף מתוך כל ה-$|V|!$ פרמוטציות ואז נבדוק אם הסידור הזה עונה על הדרישות של התכונה.

<u>פתרון</u>: M על $(< G >)$:

1. נחש סידור של קודקודי G מהצורה $(v_1,...,v_n)$.
2. בדוק לכל $i \in [1, n-1]$ שהצלע $(v_i, v_{i+1}) \in E(G)$ וגם שהצלע $(v_1, v_n) \in E(G)$.
3. אם חסרה צלע - <u>דחה</u>.
4. אחרת - <u>קבל</u>.

<u>נכונות הבניה</u>:

- אם $< G > \in L$ ← ב-G יש מעגל המילטוני מהסידור: $\{u_i\}_{i=1}^{n}$ ← באחת מהרצות המכונה ננחש את הפרמוטציה הנכונה ← כל הצלעות קיימות ← $M(G) = 1$ ← $< G > \in L(M)$.
- אם $< G > \notin L$ ← ב-G לא קיים אף סידור שיקיים מסלול המילטוני ← לכל ניחוש יהיה לפחות צלע אחת חסרה ← M ידחה לכל ריצה או לא תחזיר כלום (לא יעצור) ← $< G > \notin L(M)$.

<u>הערה</u>: הרעיון באי-דטרמיניזם זה לא לעבור על כל הפרמוטציות עד שנתפוס אחד שתקבל בשפה אלא אנחנו ננחש שרירותית רק פרמוטציה אחת בלבד (לא רצים במקביל על כמה פרמוטציות), והפרמוטציה הזאת תקבע לנו לפי הניחוש אם קיים מעגל או לא. כלומר, בשלב (1) באלגוריתם שהצגנו אנחנו מנחשים ניחוש אחד בלבד!

דוגמה 2

כדוגמה פשוטה, נסתכל על השפה: $L = \{w \in \{0,1\}^* \mid w\ starts\ and\ ends\ in\ the\ same\ bit\}$.
במודל הדטרמיניסטי, המילים $1001 \in L$ וגם $1000 \notin L$. כנ"ל במודל האי-דטרמיניסטי.
אבל, כדי לגלות כי $1000 \notin L$, נצטרך לנסות את כל מסלולי החישוב האפשריים של המכונה, ורק כשכולם דוחים אז נוכל לדעת כי $1000 \notin L$.

<u>הערות</u>:

1. מודל שקול למ"ט א"ד הוא **אי-דטרמיניזם מוגבל**: $\delta: Q' \times \Gamma \to (Q \times \Gamma \times \{L,S,R\})^2$.
2. נוח לחשוב על עץ ההרצות הזה בתור **עץ בינארי**.
3. שימו לב, שכאשר מריצים מכונה בהרצה בודדת, המכונה רצה על <u>מסלול חישוב יחיד</u> המוגדר לפי הבחירות שלה בכל הצמתים. **היא לא רצה במקביל בכל העץ!**.

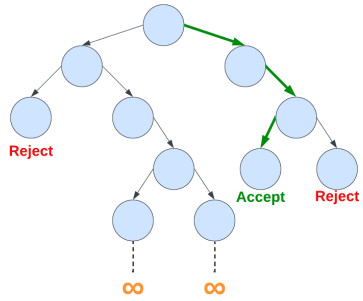
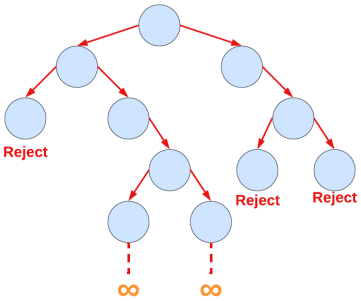
קבלת שפות ע"י מכונה לא דטרמיניסטית

<u>הגדרה</u>: מכונת טיורינג אי-דטרמיניסטית N **מקבלת** שפה L אם:

1. **לכל** $x \in L$ **קיים** מסלול מקבל בעץ החישוב של N על x.
2. **לכל** $x \notin L$ **לא קיים** מסלול מקבל בעץ החישוב של N על x. (יכולים להיות מסלולים אינסופיים).

<u>מבנה הוכחה</u>:

- אם $x \in L$ - **קיים** מסלול מקבל בעץ החישוב של המכונה N על הקלט x.
- אם $x \notin L$ - **כל** מסלולי החישוב בעץ החישוב של המכונה N על הקלט x **דוחים או אינסופיים**.

אם $x \in L(N)$ אז קיים מסלול מקבל בעץ	אם $x \notin L(N)$ אז <u>כל</u> מסלול לא מקבל
	

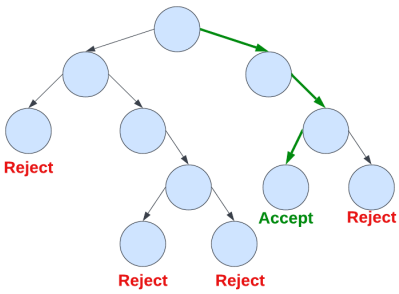
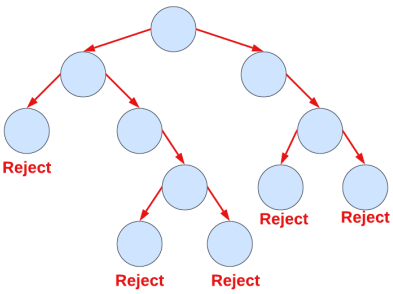
הכרעת שפות ע"י מכונה לא דטרמיניסטית

<u>הגדרה</u>: מכונת טיורינג אי-דטרמיניסטית N **מכריעה** שפה L אם:

1. **לכל** $x \in L$ **קיים** מסלול מקבל בעץ החישוב של N על x.
2. **לכל** $x \notin L$ **לא קיים** מסלול מקבל בעץ החישוב של N על x (כל המסלולים דוחים).
3. **כל המסלולים סופיים** בעץ החישוב של N, וזה נכון לכל קלט $x \in \Sigma^*$.

<u>מבנה הוכחה</u>:

- אם $x \in L$ - כל המסלולים סופיים **וקיים** לפחות מסלול אחד **מקבל**.
- אם $x \notin L$ - כל המסלולים סופיים **וכל** המסלולים **דוחים**.

אם $x \in L(N)$ אז קיים מסלול מקבל בעץ	אם $x \notin L(N)$ אז <u>כל</u> מסלול דוחה
	

דוגמה לקבלת שפה

עבור מ"ט א"ד M ומילה x נגדיר:

$$L_4 = \{(M,x) \mid M\ has\ at\ least\ 4\ different\ accepting\ paths\ when\ running\ on\ x\}$$

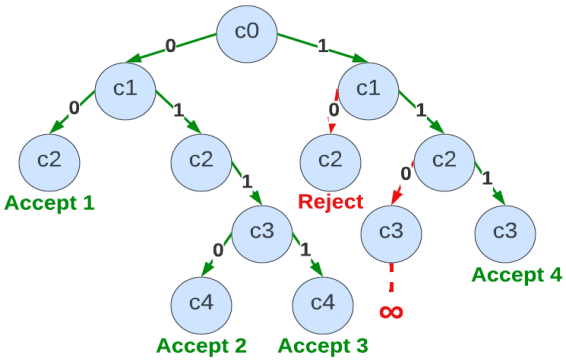
בנו מ"ט U **שתקבל** את L_4.

<u>פתרון</u>:

U על קלט $(< M >, x)$ תבצע:

1. נחש 4 מחרוזות אקראיות.
2. וודא שהן שונות זו מזו.
3. סמלץ את M על x <u>בהרצה מבוקרת</u> לפי המחרוזות האקראיות.
4. אם ב-4 הדרכים המילה התקבלה - <u>קבל</u>.

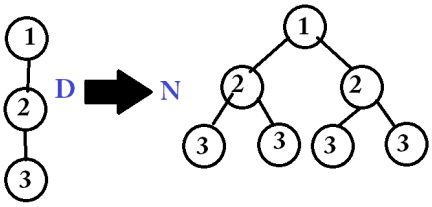
<u>הסבר</u>: נבחר 4 מחרוזות אקראיות ושונות זו מזו.
כל מחרוזת כזו תתאר לנו מסלול בעץ.
נשים לב שכל מחרוזת היא סופית ולכן אין לנו בכלל התייחסות למסלולים אינסופיים, לכן מספיק לנו רק להתייחס למקרים סופיים - מקבל או דוחה.
אנחנו נעבור על רביעיות של מחרוזות עד שנמצא 4 מחרוזות כך שכל אחת מהן מגיעה למצב מקבל עם x.
<u>הערה</u>: הקלט x לא מתאר לנו מסלול בעץ!



משפט שקילות

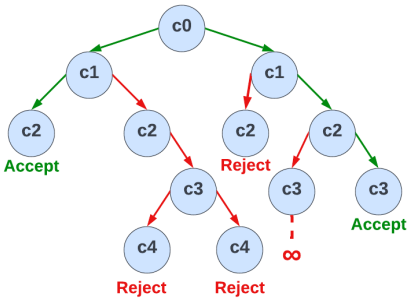
<u>משפט</u>: מכונת טיורינג לא דטרמיניסטית **שקולה בכוח החישוב שלה** למכונה דטרמיניסטית.

<u>הוכחה</u>:



- <u>כיוון א'</u>: פשוט - מכונה דטרמיניסטית D היא מקרה פרטי של מכונה לא דטרמיניסטית N. (כל האפשרויות הנוספות - זהות לאפשרות המקורית).
- <u>כיוון ב'</u>: נסביר כיצד אפשר לבנות, לכל מכונה לא דטרמיניסטית N, מכונה דטרמיניסטית D שתהיה שקולה ל-N. כלומר D תבצע סימולציה ותעקוב אחרי כל מסלולי החישוב של המכונה N.
  <u>הרעיון של כיוון ב'</u>:
  - D תנסה את כל ענפי החישוב של N על מילת הקלט w.
  - אם נמצא מסלול חישוב שמסתיים במצב המקבל (של N) אז D <u>תקבל</u> (מצב מקבל שלה).
  - אם כל מסלולי החישוב של N מסתיימים במצב הדוחה (של N) אז D <u>תדחה</u> (מצב דוחה שלה).
  - אם יש ל-N מסלולי חישוב לא עוצרים (אינסופיים) ואין לה מסלול חישוב מקבל - D <u>לא תעצור</u>.
- **בהמשך הפרק נציג את השלבים כיצד להוכיח את כיוון ב'.**

חיפוש לרוחב ולא לעומק



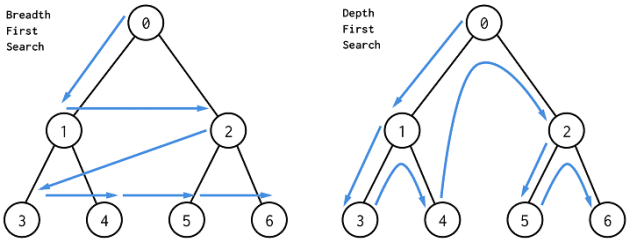
חישוב של N על w הוא **עץ** במרחב הקונפיגורציות.

- כל **צומת** בעץ הזה הוא קונפיגורציה.
- **שורש** העץ הוא הקונפיגורציה ההתחלתית c_0 של N על w.
- כל **ענף** בעץ הזה הוא חישוב חלקי אפשרי של N על w.

הערה - בחיפוש לעומק אנחנו עלולים לסרוק לעומק למסלול אינסופי.

תרגיל: הסבירו למה המכונה הדטרמיניסטית D לא תנסה לבצע חיפוש לעומק בעץ הקונפיגורציות?

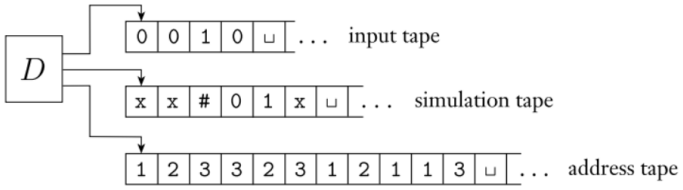
פתרון: מיד נראה שחיפוש לרוחב BFS בעץ הזה כן יהיה טוב. באופן אינטואיטיבי, אם נסתכל על הציור משמאל עבור BFS, אפשר לשער שצריך לעבור על כל קונפיגורציה אפשרית בכל רמה ולהחליט מי מביניהם יכול להתאים.



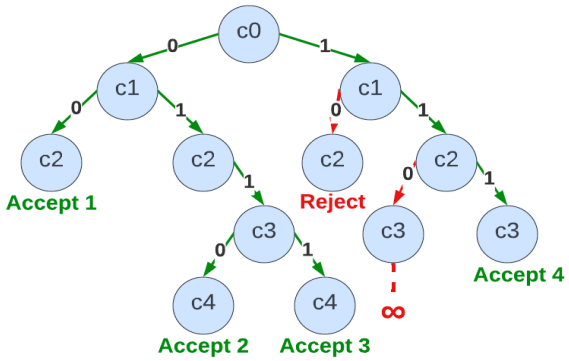
המכונה הדטרמיניסטית

למכונה D יהיו **שלושה סרטים**.

1. **סרט הקלט** - שעליו רשומה מילת הקלט. זהו סרט לקריאה בלבד (לא משנים בו דבר).
2. **סרט הסימולציה** - לחיקוי מסלול חישוב חלקי של N. נעדכן בו את הקונפיגורציות.
3. **סרט הכתובות** - שיציין את מסלול הבחירות הלא דטרמיניסטיות של החישוב החלקי הנוכחי.



סרט הכתובות



לכל צומת פנימית c_i (לכל קונפיגורציה) בעץ החישוב יש לכל היותר 2 בנים, לכן מילה $w \in \{0,1\}^*$ יכולה לציין מסלול חישוב חלקי מהשורש עד הצומת c_i בעץ. לכן, המילה w היא הכתובת של הצומת בעץ. לדוגמה בציור משמאל המחרוזת $w = 011$ תביא אותנו מהשורש אל צומת פנימית.

כתובות לא חוקיות: מאחר ולקונפיגורציה ההתחלתית c_0 יש רק שתי קונפיגורציות עוקבות (כי לשורש העץ יש שני בנים), אז כל כתובת שמתחילה ב-2 ומעלה איננה חוקית. כלומר אם אמרנו שהמחרוזות מורכבות מ-$w \in \{0,1\}^*$ אז לא יכול להיות לנו מחרוזת למשל מהצורה $w = 0112230110$.

פתרון: המכונה D תתייחס לכתובת לא חוקית כאל ענף חישוב שהסתיים בדחייה.

הסימולציה של D

תזכורת: אנחנו צריכים להוכיח את כיוון ב' - לכל מכונה לא דטרמיניסטית N נבנה מכונה דטרמיניסטית D ששקולה ל-N.

אתחול: בתחילת הריצה של D.

1. מילת הקלט רשומה על סרט 1 שהוא סרט הקלט.
2. סרט הסימולציה (סרט 2) וסרט הכתובות (סרט 3) ריקים (יש $\bar{b}$ בכל התאים בסרטים האלה).

שלב בסימולציה:

1. מעתיקים את תוכן סרט 1 (הקלט) לסרט 2.
2. מבצעים את הצעדים של N על סרט 2 לפי הבחירות הלא דטרמיניסטיות הרשומות בסרט 3 (לפי הכתובות). (כלומר מעדכנים את הקונפיגורציות לרוחב על סרט 2).
3. אם N הגיעה למצב המקבל שלה, מקבלים. כלומר D נכנסת למצב המקבל שלה, והחישוב מסתיים.
4. אחרת, עוברים לבצע סימולציה של ענף החישוב הבא.

הסברים:

- מה פירוש "אחרת"? מה יכול לקרות?:
  - יכול לקרות ש-D סיימה לבצע את הצעדים שהיו רשומים בכתובות ולא הגיעה למצב עצירה.
  - ייתכן כי N נכנסה למצב הדוחה שלה באיזשהו צעד בענף החישוב הנוכחי.
  - ייתכן שהכתובת בסרט השלישי לא חוקית (למשל מכילה '3' אבל המחרוזת צריכה להיות בינארית).
- איך עוברים לענף החישוב הבא?:
  - מוחקים את תוכן סרט הסימולציה.
  - עוברים לכתובת הבאה בסרט הכתובות.
  - מתחילים את הסימולציה של ענף החישוב הבא.

סיכום הוכחת כיוון ב' של השקילות שעשינו

- אם $x \in L(N)$ קיים לפחות מסלול אחד מקבל בעץ החישוב של N על x בזמן $N(x) = 1$ סופי, D תסרוק את העץ עד לרמת הקונפיגורציה המקבלת בזמן סופי, הגיעה לקונפיגורציה מקבלת ותקבל $D(x) = 1$ כלומר $x \in L(D)$.
- אם $x \notin L(N)$ כל המסלולים דוחים או אינסופיים מקרה א': אם כל המסלולים דוחים אז בשלב מסויים נסיים לסרוק את העץ, לא נגיע למצב מקבל ולכן נדחה. מקרה ב': אם יש מסלול אחד אינסופי והיתר דוחים אז נמשיך לרוץ על המסלול האינסופי ולא נקבל דבר. בשני המקרים $x \notin L(D)$.

מסקנה מן השקילות

1. כדי להוכיח ש: $L \in RE$ אפשר לתאר מכונה לא דטרמיניסטית שמקבלת את L.
   משום שלכל מכונה כזו אפשר לבנות מכונה דטרמיניסטית שתקבל את L.
2. כדי להוכיח ש: $L \in R$ אפשר לתאר מכונה לא דטרמיניסטית שמכריעה את L.
   עץ החישוב במקרה זה יהיה סופי לכל קלט.

הערה על זמן הריצה:
המכונה D שתיארנו **לא יעילה** מבחינת זמן הריצה כי יש בה חזרות על חלקים של חישובים.
אפשר לייעל את פעולתה של D, אך עדיין זמן הריצה של כל סימולציה מוכרת של מכונה לא דטרמיניסטית ע"י מכונה דטרמיניסטית הוא **לפחות מעריכי (אקספוננציאלי)** בזמן הריצה של המכונה הלא דטרמיניסטית.
כלומר, תהליך הסימלוץ של D על N הוא יקר בזמן הריצה, כלומר הוא **לפחות** בזמן אקספוננציאלי.
מכונה דטרמיניסטית מכריעה - זמן הריצה הוא מספר הקונפיגורציות בסדרה עד לקונפיגורציה הסופית.
מכונה לא דטרמיניסטית מכריעה - זמן הריצה של N על מילה x מוגדר כמספר הצעדים במסלול החישוב **הארוך ביותר בעץ** עבור N על x.

שימושי המודל הלא דטרמיניסטי

למה בכלל שנרצה להשתמש במודל לא דטרמיניסטי? הרי הוא לא מציאותי ולא שימושי וגם מספר המסלולים הדוחים יכול להיות גדול. **אחד היתרונות** לשימוש במודל א"ד הוא היכולת לנחש מחרוזות וככה לחסוך בהרצה מבוקרת. אפשר לנחש בנוסף מסלול ריצה.

לדוגמה: נתונה השפה $L = \{< M > : L(M) \neq \emptyset\}$.

- במודל הדטרמיניסטי היינו צריכים לעשות הרצה מבוקרת חזקה, כלומר לעבור על כל המילים ולהבטיח שקיימת מילה שמתקבלת בשפה.
- במודל הלא דטרמיניסטי אפשר לנחש מילה x, לסמלץ אותה על M, אם היא התקבלה אז תקבל ואחרת תדחה. לכן אם קיים מסלול בעץ החישוב שיקבל את המילה x אז שפת המכונה לא ריקה.

דוגמה 1 - לשימוש במכונה לא דטרמיניסטית

הקדמה: פונקציה של מ"ט M היא f_M. נתונה השפה: $L = \{< M > : \exists x \in \Sigma^*, f_M(x) = x\}$.
כלומר שקיימת לפונקציה של המכונה נקודת שבת. אנחנו רוצים ליצור מכונה א"ד לשפה הזאת.

- כדי להוכיח $L \in RE$ במודל הדטרמיניסטי היינו צריכים לבצע הרצה מבוקרת חזקה של כל המילים בסיגמא כוכבית.
- לעומת זאת, בעזרת ניחוש, אפשר לחסוך את ההרצה המבוקרת החזקה הזאת.
  כלומר, אנחנו נרצה לנחש $x \in \Sigma^*$ שעבורו נפעיל את M ונקבל בהתאם לפלט של M.

<u>הוכחה</u>: נרצה להוכיח כי $L \in RE$ ולכן נבנה מכונת טיורינג לא דטרמיניסטית N <u>המקבלת</u> את L:
המכונה N על הקלט $< M >$ תבצע:

1. נחש מילה $x \in \{0,1\}^*$
2. סמלץ את M על x:
   a. אם $f_M(x) = x$ אז <u>קבל</u>.
   b. אם $f_M(x) \neq x$ אז <u>דחה</u>.

נוכיח את נכונות המכונה N, כלומר נוכיח כי $L(N) = L$:

- אם $< M > \in L$ אז קיים (לפי ניחוש שרירותי) מילה $x \in \{0,1\}^*$ כך ש: $f_M(x) = x$. בעץ החישוב בהפעלת המכונה N על הקלט $< M >$ קיים מסלול שבו הניחוש של N הוא בדיוק $x \leftarrow$ המסלול המדובר הוא **מסלול מקבל** $\leftarrow < M > \in L(N)$.
- אם $< M > \notin L$ אז **לכל** $x \in \{0,1\}^*$ מתקיים $f_M(x) \neq x \leftarrow$ בעץ החישוב בהפעלת המכונה N על הקלט $< M >$ לכל $x \in \{0,1\}^*$ **לא קיים** מסלול שבו הניחוש של N הוא בדיוק $x \leftarrow$ כל המסלולים של N הם **דוחים או אינסופיים** $\leftarrow < M > \notin L(N)$.

∎

הערה חשובה

- פונקציית הרדוקציה היא **תמיד** דטרמיניסטית!
- יכול להיות שבפונקצית הרדוקציה תהיה מכונה א"ד אבל הפונקציה עצמה היא דטרמיניסטית.
- תוצאת פונקצית הרדוקציה יכולה להיות קידוד של מ"ט א"ד.

דוגמה 2 - לשימוש במכונה לא דטרמיניסטית

מודל נוסף למ"ט א"ד:

$$L_4 = \{< M, x > \mid M \text{ has at least 4 different accepting paths when running on } x\}$$

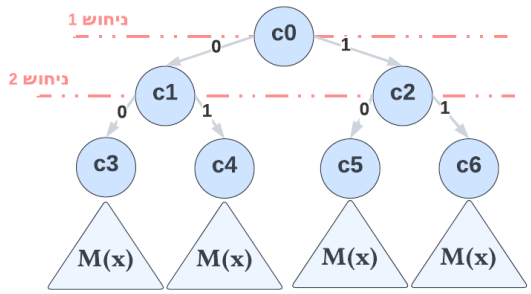
$$L_u = \{< M, x > \mid M \text{ accepts } x\}$$

הוכיחו שקילות בין מודלים.

<u>פתרון</u>: נרצה לעשות רדוקציה $L_u \leq L_4$ ואחר כך נעשה גם הפוך $L_4 \leq L_u$.

<u>הוכחה כיוון א'</u>:

- <u>אינטואיציה</u>: אנחנו רוצים שאם מכונה M תקבל את x אז יהיה לנו מכונה אחרת M' שתקבל את x 4 פעמים שונים. בתיאור הרדוקציה, אנחנו נרצה לשכפל את המסלולים, כלומר נשכפל פעמיים כאשר כל שכפול פותח לנו רמה חדשה בעץ החישוב, לכן עבור 2 שכפולים נקבל $2^2 = 4$



צמתים - כלומר צמתים סופיים ונרצה שמכל הארבעה אלו ימשיך מסלול המקבל את x על M.

- <u>תיאור הרדוקציה</u>: נראה ברדוקציה $L_u \leq L_4$.
  - פונקצית הרדוקציה היא: $f(< M, x >) =< M', y >$,
  - המכונה M' על הקלט y:
    - תנחש 0 או 1
    - תנחש 0 או 1
    - סמלץ את M על x וענה כמותו.
- <u>הוכחת נכונות פונקציית הרדוקציה</u>:
  - הפונקציה מלאה: לכל M, x ניתן ליצור M', y.
  - הפונקציה ניתנת לחישוב: לכל מ"ט קיים קידוד במיוצג כמחרוזת.
  - הפונקציה פתירה:
    - $< M, x >\in L_u \leftarrow x \in L(M) \leftarrow$ קיים לפחות מסלול אחד מקבל בחישוב M על x
      $\leftarrow$ קיימים 4 מסלולים מקבלים $< M', x >\in L_4$.
    - $< M, x >\notin L_u \leftarrow x \notin L(M) \leftarrow$ כל מסלול בחישוב M על x או דוחה או אינסופי
      $\leftarrow$ קיימים 4 מסלולים דוחים או אינסופיים $< M', x >\notin L_4$.
- <u>לסיכום</u>: הראנו רדוקציה L_u מעל L_4 והוכחנו ש: $L_4 = L(M')$.

<u>הוכחה כיוון ב'</u>: נראה ברדוקציה $L_u \leq L_4$. **להשלים**.

הוכחת שקילות בין המודלים

זה גרסה שלא נלמדה הסמסטר אבל למדו אותה בשנים קודמות והיא יותר מפשטת - (לא מתקבל במבחן).

<u>טענה</u>: מודל מ"ט א"ד שקול מבחינה חישובית למ"ט דטרמיניסטית.

כלומר, תהי N מ"ט א"ד , אזי קיימת מ"ט דטרמיניסטית D כך שמתקיים: $L(D) = L(N)$.

<u>הוכחה</u>:

- ניתן לייצג את ריצת N על w כעץ העובר בין הקונפיגורציות של המכונה (יתכן שהעץ הוא אינסופי)
- נתאר את D על קלט w:
  - סרוק לרוחב את עץ החישוב של N על w בעזרת BFS.
  - אם זיהית קונפיגורציה מקבלת - <u>קבל</u>.
  - אם סרקת את כל העץ ולא מצאת קונפיגורציה מקבלת - <u>דחה</u>.
- נכונות - אם קיים מסלול מקבל בריצת N על w, אזי D מגיע בסופו של דבר לקונפיגורציה מקבלת ותקבל את w. אחרת תדחה.

שאלה 1 - תרגול 4 (שאלת מבחן מועד א' - קיץ 2019)

בהינתן מכונת טיורינג אי-דטרמיניסטית M, נגדיר: המכונה ההופכית של M, הינה מכונה הזהה בפעולתה לפעולת M, למעט החלפת המצב המקבל והמצב הדוחה אחד בשני. נסמן את המכונה ההופכית ב-$M_{inverse}$.

סעיף א': הוכיחו או הפריכו: $\overline{L(M)} = L(M_{inverse})$ (השפה ההופכית של שפת המכונה M, שווה לשפת המכונה ההופכית).

סעיף ב': האם קיימת מכונה M, כך שמתקיים: $L(M) = L(M_{inverse})$? אם כן, תנו דוגמא, אם לא, הוכיחו.

פתרון סעיף א':

- אינטואיציה: המכונה אי דטרמיניסטית ולכן יכול להיות שעבור x מסוים מהשפה נקבל מסלול מקבל ומצד שני יהיה עבורו מסלול אחר שהוא דוחה. לכן זה הפרכה - צריך לתת דוגמה נגדית.
- הפרכה:
  - נניח שקיים x כך שב-M על x ישנו מסלול מקבל ומסלול דוחה. מכאן ש: $x \in L(M)$ כלומר $x \notin \overline{L(M)}$.
  - נתבונן בריצת המכונה $M_{inverse}$ על x:
    - עבור המסלול הדוחה של M על x - $M_{inverse}$ יקבל את x.
    - עבור המסלול המקבל של M על x - $M_{inverse}$ ידחה את x.

    בכל אופן, $x \in L(M_{inverse})$ (כי קיים לו מסלול מקבל).
  - מכאן ש: $L(M_{inverse}) \neq \overline{L(M)}$ כי הראנו ש: $x \in L(M)$ וגם ש: $x \in L(M_{inverse})$.
- שאלת צד: מתי זה נכון? - כאשר כל המצבים סופיים וזהים (כולם מקבלים או כולם דוחים).

פתרון סעיף ב':

- קיים מכונה כזאת.
- אופציה א': נדאג שבמכונה שלנו על כל x קיימים מסלולים גם מקבלים וגם דוחים.
  - מצד אחד: $L(M) = \Sigma^*$.
  - מצד שני עבור $M_{inverse}$ קיימים מצבים מקבלים לכל x (המצבים שדחו קודם) כלומר $L(M_{inverse}) = \Sigma^*$.

  בכל מקרה, $L(M) = L(M_{inverse}) = \Sigma^*$. (כי אם קיימים מסלולים מקבלים וגם דוחים אז שנהפוך מקבל מסלולים דוחים וגם מקבלים, ככה או ככה קיים לנו לפחות מסלול מקבל אחד).
- אופציה ב': מכונה שעל כל x בכל המסלולים נכנסת ללולאה.
  - מצד אחד: $L(M) = \emptyset$.
  - מצד שני: $L(M_{inverse}) = \emptyset$.

  בכל מקרה, $L(M) = L(M_{inverse}) = \emptyset$. (כי ההפך של "לא מוגדר" הוא "לא מוגדר").

שאלה 2 - תרגול 4

נתונה השפה $L = \{(w_1, w_2, ..., w_n) \mid \exists i \; s.t. \; w_i = i\}$.

הוכיחו כי $L \in R$.

<u>אינטואיציה</u>: אם היינו עובדים עם מכונה דטרמיניסטית אז המכונה הייתה עוברת בלולאה לינארית מ-1 עד n ובודקת עבור כל צעד האם $w_i = i$. מצד שני, אם נעבוד עם מכונה אי-דטרמיניסטית אז אפשר לנחש ולכן זה יותר קל למימוש כי לא צריך לולאות. אנחנו בונים על זה שהניחוש יהיה נכון.

<u>הוכחה</u>:

<u>אלגוריתם</u>: נתאר מכונה א"ד M <u>המכריעה</u> את השפה L.

M על קלט $(w_1, w_2, ..., w_n)$:

1. נחש אינדקס $1 \leq i \leq n$:
   a. אם $w_i = i$ - <u>קבל</u>.
   b. אחרת, <u>דחה</u>.

<u>הוכחת נכונות</u>:

- $(w_1, w_2, ..., w_n) \in L$ ← **קיים** i שעבורו $w_i = i$ ← **קיים** ניחוש (מסלול חישוב מקבל) שעבורו M **מקבל** ← $(w_1, w_2, ..., w_n) \in L(M)$.
- $(w_1, w_2, ..., w_n) \notin L$ ← **לא קיים** אף i שעבורו $w_i = i$ ← **לכל** ניחוש (מסלול חישוב מקבל) שעבורו M **דוחה** ← $(w_1, w_2, ..., w_n) \notin L(M)$.

∎

שאלה 3 - תרגול 4

נתונה מכונה א"ד:

<u>אלגוריתם</u>: N על קלט $(< M >, x)$:

1. מנחשת מספר n.
2. מריצה את M על x למשך n צעדים.
   a. אם M עצרה - אז N <u>תקבל</u>.
   b. אם M לא עצרה - אז N <u>תדחה</u>.

<u>השאלה</u>: מהי השפה של N?.

<u>תשובה</u>: $L(N) = HP$.

<u>הסבר</u>: ההגבלה פה לא קשורה בצעדים, כי אם היא מקבלת אז קיים n מספיק גדול כדי שיהיה מספיק צעדים לרוץ ואם היא דוחה אז לא משנה כמה n מספיק גדול - היא לא תעצור לעולם.

<u>הוכחת נכונות</u>:

- $(< M >, x) \in HP$ ← M עוצרת על x ← נניח ש: M עוצרת על x לאחר t צעדים ← לכל ניחוש של $n \geq t$ כך ש: n, המכונה N תעצור ותקבל. ← קיים מסלול חישוב שבו N תקבל ← $(< M >, x) \in L(N)$.
- $(< M >, x) \notin HP$ ← M לא עוצרת על x ← לא עוצרת על x לכל ניחוש n ← לא קיים אף מסלול חישוב שבו N תקבל ← $(< M >, x) \notin L(N)$.

∎

שאלה 4 - תרגול 4

נתונה השפה: $COMPOSITE = \{n \mid n\ is\ composite\}$.

הוכיחו: $COMPOSITE \in R$.

<u>תזכורת</u>: מספר פְּרִיק הוא מספר שלם חיובי שאפשר לכתוב אותו כמכפלה של שני שלמים גדולים מ-1. מספרים אלה נקראים 'גורמים' של המספר הנתון. כל מספר שלם גדול מ-1 הוא או ראשוני או פריק.

לדוגמה, המספר 14 הוא פריק מכיוון שאפשר לפרק אותו כמכפלה של 2 ו-7, ולכן 2 ו- 7 הם הגורמים של 14.

<u>הוכחה</u>: נתאר מ"ט א"ד M <u>המכריעה</u> את השפה $COMPOSITE$.

<u>אלגוריתם</u>: M על קלט n (או שניתן לסמן $M(n)$) תבצע:

1. תנחש מספר k כך ש: $1 < k < n$.
2. אם $k\%n == 0$ אז M <u>תקבל</u>.
3. אחרת - M <u>תדחה</u>.

<u>נכונות הבניה</u>:

- $n \in COMPOSITE$ ← קיים k שמחלק את n ← קיים ניחוש של M עבורו יקבל ← $n \in L(M)$.
- $n \notin COMPOSITE$ ← לא קיים אף k שמחלק את n ← לכל ניחוש של k כזה, M תדחה ← $n \notin L(M)$.

∎

שאלה 5 - תרגול 4

נתונה מכונה M_0 ונתון קבוע טבעי $c \in \mathbb{N}$.

נתונה השפה: $L_{M_0,c} = \{< M > \mid L(M) = L(M_0)\ and\ |< M >| = c\}$.

<u>שאלה</u>: באיזו מחלקה חישובית השפה נמצאת?

<u>תשובה</u>: $L_{M_0,c} \in R$.

<u>הסבר</u>: למה זה ב-R?. בפרק של BHP הראנו שפות שאם יש להם חסם אז אפשר להכריע אם הקלט עקף את החסם או לא וככה להכריע. מאחר ויש מספר סופי של מחרוזות בינאריות באורך c, כלומר 2^c מחרוזות האלה, אז זו שפה סופית ← רגולרית ← שייכת ל-R.

- נקודה חשובה: אם היה בשפה or במקום and אז זה היא לא הייתה סופית ובכלל לא ב-R.

שאלה 6 - תרגול 4 (שאלת מבחן)

נתונה שפה:

$$L = \{< M, x > \mid \exists\ a\ Turing\ Machine\ M',\ s.t.\ M'\ accepts\ x\ or\ M\ accepts\ < M' >\}$$

<u>שאלה</u>: באיזו מחלקה השפה?.

<u>תשובה</u>: $L \in R$. למעשה זו Σ^* כי ניתן תמיד לבחור $M' = M_{STAM}$ שתקבל את x.

<u>הסבר</u>: אכן <u>קיימת</u> מכונה M' שיכולה לקבל כל קלט שהיא מקבלת, זוהי המכונה M_{STAM}. גם אם התנאי השני (צד ימין של ה-or) לא מקבל, זה בסדר כי M_{STAM} תמיד תקבל את x. מכונה שמקבלת כל קלט זו מכונה שמכריעה את השפה Σ^*.

שאלה 7 - תרגול 4 (שאלת מבחן)

נתונה שפה (אותו שפה כמו שאלה 6 אבל עם **and** במקום or):

$$L = \{< M, x > \mid \exists\ a\ Turing\ Machine\ M',\ s.t.\ M'\ accepts\ x\ and\ M\ accepts\ < M' >\}$$

<u>שאלה</u>: באיזה מחלקה השפה?.

<u>תשובה</u>: $L \in RE$.

- <u>אלגוריתם</u>: נתאר מ"ט א"ד N שמקבלת את L.

  N על קלט $< M, x >$:

  1. <u>תנחש</u> מכונה M'.
  2. תסמלץ מכונה M' על x, אם M' דחתה את x - N <u>דוחה</u>.
  3. תסמלץ מכונה M על $< M' >$, אם M דחתה את $< M' >$ - N <u>דוחה</u>.
  4. אחרת, N <u>מקבלת</u>.

- <u>נכונות הבניה</u>:
  - $< M, x > \in L$ ← קיים מכונה M' שעונה על דרישות השפה ← N תנחש את M' עבורו קיים מסלול חישוב מקבל ← N תקבל ← $< M, x > \in L(N)$.
  - $< M, x > \notin L$ ← לא קיים מכונה M' שעונה על דרישות השפה ← לכל ניחוש (מסלול חישוב), N תדחה או לא תעצור ← $< M, x > \notin L(N)$.

<u>שיעורי בית</u>: הוכיחו שהשפה לא ב-R. (ברדוקציות - אפשר מ-HP או מקור אחר).

שאלה 8 - תרגול 4

נתונה השפה (כאשר N היא מכונה א"ד):

$$L = \{< N, x > \mid N\ running\ on\ x\ has\ an\ accepting\ path\ and\ a\ rejecting\ path\}$$

הוכיחו שהשפה $L \notin R$.

<u>פתרון</u>:

- <u>אינטואיציה</u>: השפה ב-RE. נוכיח באינדוקציה ש:$L \notin R$ ונשתמש בשפה $HP \notin R$. בתיאור הרדוקציה נפעיל את M על x, אם הוא עצר (לפי HP) אז הגענו במסלול החישוב שלנו לצומת שבה צריך לקבוע מה הצעד הבא של האלגוריתם - האם נקבל או נדחה? לכן, מאחר ש-N א"ד אז נגריל/ננחש את הענף הבא במסלול החישוב שיקבע אם נקבל או נדחה (0 == דוחה || 1 == מקבל).
- <u>תיאור הרדוקציה</u>: נראה רדוקציה $HP \leq L$.
  - פונקציית הרדוקציה: $f(< M, x >) =< N, y >$.
  - N על קלט y תפעל:
    - תסמלץ את M על x.
    - מגרילה ביט b.
    - אם $b = 0$ אז N <u>דוחה</u>.
    - אם $b = 1$ אז N <u>מקבלת</u>.
- <u>הוכחת נכונות הפונקציה</u>:
  - הפונקציה מלאה: לכל M, x ניתן ליצור N, y.
  - הפונקציה ניתנת לחישוב: לכל מ"ט קיים קידוד.
  - הפונקציה פתירה:
    - $< M, x > \in HP \leftarrow M$ עוצרת על $x \leftarrow N$ יכולה לקבל או לדחות ← יש ל-N מסלול חישוב מקבל ומסלול חישוב דוחה ← במ"ט א"ד מספיק מסלול מקבל אחד בשביל לקבל את המילה ← $< N, x > \in L(N)$.
    - $< M, x > \notin HP \leftarrow M$ לא עוצרת ← אין ל-N מסלול חישוב מקבל ← $< N, x > \notin L(N)$.
- <u>סיכום</u>: מאחר ו-$HP \notin R$ והראנו ברדוקציה $HP \leq L$ אזי גם $L \notin R$.

∎

תרגילים - חישוביות

<u>תרגיל 1</u>: כתבו מכונת טיורינג המחשבת את הפונקציה $f(x) = 2x$.

הערה: x הוא מחרוזת בינארית של מספר חיובי. תזכורת: הכפלה בינארית ב-2^i דורש הזזת i צעדים שמאלה.

<u>תרגיל 2</u>: מודל מכונת טיורינג בגיל העמידה הוא מודל זהה למודל הרגיל של מ"ט לחישוב פונקציות פרט לשינויים הבאים:

1. למכונה אין מצבים סופיים, ולכן המכונה לעולם לא עוצרת.
2. נגיד שמכונה נעמדת בצעד החישוב ה-t אם מצעד חישוב זה והלאה המכונה מבצעת אך ורק תזוזה $Stay$ (ולעולם לא תבצע יותר תזוזות $Left$ או $Right$).
3. עבור קלט x הפלט של המכונה הוא המילה y שנמצאת משמאל לראש כאשר המכונה נעמדת.

הוכיחו/הפריכו: מודל גיל העמידה שקול למודל הרגיל.

<u>תרגיל 3</u>: כתבו מכונת טיורינג שמחשבת את הפונקציה: $f(w) = w^R$.

<u>תרגיל 4</u>: הוכיחו כי RE סגורה לחיתוך.

<u>תרגיל 5</u>: יהיו $L_1, L_2 \in RE \backslash R$ האם ייתכן ש:

1. $L_1 \cap L_2 \in R$.
2. $L_1 \cup L_2 \in R$.

קבעו את תשובתכם, במידה וכן הראו שפות כאלה. (שאלה אונ' תל אביב מטלה 5, 2006a).

<u>תרגיל 6</u>: קבעו עבור השפות הבאות האם שייכות ל- $(R,\ coRE,\ RE)$. הוכיחו.

$$L_1 = \{M \mid \exists x\ s.t.\ M\ halts\ on\ x\}$$

$$L_2 = \{M \mid |L(M)| \leq 3\}$$

(שאלה אונ' תל אביב מטלה 3, 2008a).

<u>תרגיל 7</u>: שאלת מבחן - מועד ב', קיץ 2020

לכל אחד מהסעיפים הבאים, קבעו האם נכון או לא נכון.

אם הטענה נכונה - הסבירו. אם טענה אינה נכונה - תנו דוגמה נגדית:

א. לכל שפה $L \in R$ מתקיים $L \leq HP$ וגם $L \leq \overline{L_u}$.

ב. אם $L_1 \in RE \backslash R$ וגם $L_2 \in coRE \backslash R$ אז מתקיים $L_1 \cup L_2 \notin R$.

ג. אם $L_1 \in RE$ וגם $L_2 \in coRE$ אז מתקיים $L_1 \cap L_2 \in R$.

<u>תרגיל 8</u>: שאלת מבחן - מועד א', קיץ 2020

לכל אחת מהשפות הבאות, קבעו האם היא ב-R? והאם היא ב-RE? הוכיחו את תשובתכם.

א. $L_1 = \{< M, x_1, x_2 > \mid M \ accepts \ x_1 \ and \ rejects \ x_2\}$.

ב. $L_3 = \{< M_1 >< M_2 > : \left|L(M_1) \cap L(M_2)\right| = \infty\}$.

<u>תרגיל 9</u>: שאלת מבחן - מועד א', קיץ 2020

הוכיחו או הפריכו את הטענות הבאות:

א. אם $L \in R$ וגם $L \neq \varphi, \Sigma^*$ אז $L \leq \bar{L}$.

ב. אם $L \notin R$ וגם $L \leq \bar{L}$ אז $L \in RE \cup coRE$.

<u>תרגיל 10</u>: קבעו עבור השפות הבאות האם שייכות ל- $(R,\ coRE,\ RE)$. הוכיחו.

$$L_1 = \{< M > \mid when \ M \ running \ \varepsilon, \ the \ head \ moves \ left \ at \ most \ 2022 \ times\}$$

$$L_2 = \{< M > \mid when \ M \ running \ \varepsilon, \ the \ head \ never \ moves \ three \ times \ in \ a \ row \ left\}$$

(שאלה אונ' תל אביב מטלה 5, 2014b).

<u>תרגיל 11</u>: הוכיחו או הפריכו: קיימת שפה שלמה ב-$coRE$.

<u>תרגיל 12</u>: עבור השפה הבאה ענו האם היא ב-R והאם היא ב-RE.

$$L = \{< M > \mid M \ halts \ on \ all \ inputs\}$$

(מטלה 2 - אריאל תש"ף).

<u>תרגיל 13</u>: קבעו לאיזה מחלקות $(R,\ coRE,\ RE)$ השפות הבאות שייכות:

$$L_1 = \{< M > \mid M \ halts \ on \ at \ least \ one \ input\}$$

$$L_2 = \{< M > \mid M \ halts \ on \ at \ most \ one \ input\}$$

(שאלת מבחן אריאל מועד ב' תשע"ז).

<u>תרגיל 14</u>: תהיינה L_1, L_2 שפות כך ש: $L_1 \in RE$ וגם $L_2 \in coRE$ בנוסף נתון כי $L_1 \leq L_2$.

עבור כל אחת מהטענות הבאות הוכיחו האם היא תמיד נכונה, תמיד לא נכונה או שקיימות שפות המקיימות את התנאים עבורן הטענה נכונה, וקיימות שפות המקיימות את התנאים עבורן הטענה לא נכונה.

1. L_1 היא RE משלימה.
2. $L_2 \in R$. כלומר L_2 כריעה.

3. $L_1 \notin R$, כלומר L_1 לא כריעה.

(מבחן אריאל תשע"ז קיץ מועד א שאלה 4).

<u>תרגיל 15</u>: הראו ש: $\{< M > | |L(M)| \geq 6\} \leq \{< M > | |L(M)| \geq 3\}$.

(מבחן אריאל תשע"ח קיץ מועד א' שאלה 2ג').

מבוא לסיבוכיות

מבוא

חישוביות: בחלק הקודם, התעסקנו בשאלה "מה בכלל אפשר לחשב בעזרת מכונות?".

- איזה פונקציות ניתנות לחישוב ואיזה לא?
- איזה שפות ניתנות להכרעה (לזיהוי) ואיזה לא?

סיבוכיות: בחלק זה, נעסוק:

- בפונקציות הניתנות לחישוב ובשפות הכריעות.
- בשאלה של כמויות המשאבים הדרושות לחישובים (משאבי זמן וזיכרון הדרושים).

סיבוכיות זמן

בסיבוכיות זמן נשאל מהו הזמן הדרוש למימוש כל אלגוריתם.

המודל החישובי שאיתו נמדוד את סיבוכיות הזמן הוא מכונת טיורינג והסיבות לכך הן:

1. סיבוכיות הזמן נמדדת כפונקציה של גודל הקלט ובמכונת טיורינג גודל הקלט מוגדר היטב (מספר התאים בסרט).
2. זמן הריצה יוגדר כמספר הצעדים שמבצעת המכונה במהלך ריצתה (צעד החישוב במ"ט מוגדר היטב).

דוגמה 1 - לחישוב זמן ריצה

כמה זמן נדרש להכריע שייכות לשפה $L_1 = \{a^m b^m | m \geq 0\}$ במ"ט חד-סרטית (סרט אחד בלבד)?

האלגוריתם: נתאר את M_1 על מילת הקלט w:

1. עבור על הקלט עד סופו, ודחה אם יש a מימין ל-b.
2. על עוד יש a-ים ו-b-ים, עבור על תוכן הסרט, ומחק a אחד ו-b אחד בכל מעבר כזה.
3. אם נמחקו כל ה-a-ים וכל ה-b-ים, קבל. אחרת, דחה.

זמן הריצה: זמן הריצה של M_1 הוא $O(n^2)$ כאשר n הוא גודל הקלט.

הגדרה - זמן הריצה במקרה הגרוע ביותר

הגדרת זמן הריצה במקרה הגרוע ביותר: תהי M מ"ט דטרמיניסטית, ותהי פונקציה $t: \mathbb{N} \to \mathbb{N}$.

נאמר שזמן הריצה של המכונה M הוא $t(n)$ אם לכל קלט w באורך n, מספר צעדי הריצה של M על w הוא לכל היותר $t(n)$.

לדוגמה: בדוגמה הקודמת אפשר להגיד שמספר צעדי הריצה של M על w הוא לכל היותר $O(t(n)) = O(n^2)$. כי אמנם גודל הקלט הוא n אבל נדרש מהמכונה M_1 לכל היותר n^2 צעדי ריצה.

הגדרה - מחלקה של שפות

<u>הגדרה - מחלקה של שפות</u>: תהי פונקציה $t: \mathbb{N} \to \mathbb{N}$.

נסמן ב-$DTIME(t(n))$ את <u>מחלקת השפות</u> שיש להן מכונת טיורינג <u>דטרמיניסטית מכריעה</u> שזמן הריצה שלה הוא $O(t(n))$.

פורמלית: $DTIME(t(n)) = \{L \mid L\ is\ a\ language,\ decided\ by\ an\ O(t(n)) - time\ DTM\}$.

<u>דוגמה</u>: השפה $L_1 = \{a^m b^m | m \geq 0\}$ שייכת למחלקה $DTIME(n^2)$, כלומר $L_1 \in DTIME(n^2)$.

דוגמה 2 - לחישוב זמן ריצה

האם אפשר להכריע את השפה מדוגמה 1: $L_1 = \{a^m b^m | m \geq 0\}$ באופן מהר יותר?

כלומר, האם קיים אלגוריתם שפותר את הבעיה בזמן פחות מ-$O(n^2)$? - כן.

<u>אלגוריתם</u>: M_2 על מילת קלט w תבצע:

1. עבור על הקלט w, אם יש a מימין ל-b אז <u>דחה</u>.
2. חזור על הפעולות הבאות כל עוד יש a-ים ו-b-ים:
   a. עבור על תוכן הסרט: אם מספר כל ה-a-ים וה-b-ים הוא אי זוגי, אז <u>דחה</u>.
   b. עבור על תוכן הסרט: מחק את ה-a הראשון, השלישי, החמישי וכך הלאה, וכך גם ביחס ל-b-ים.
3. אם נמחקו כל ה-a-ים וה-b-ים, אז <u>קבל</u>.
4. אם לא נמחקו כל ה-a-ים וה-b-ים, אז <u>דחה</u>.

aaaaabbbbb
~~a~~a~~a~~a~~ab~~b~~b~~b~~b~~b
aabb
~~a~~a~~b~~b
ab
~~ab~~

<u>זמן הריצה</u>: זמן הריצה של המכונה M_2 היא:

- <u>שלב 1</u>: מעבר על כל הקלט עולה לנו $O(n)$ צעדים.
- <u>שלב 2</u>: נכנסים ללולאה שתבצע לכל היותר $O(\log(n))$ צעדים.
  - <u>שלב a</u>: עוברים על תוכן הסרט שעולה $O(n)$ צעדים.
  - <u>שלב b</u>: עוברים על תוכן הסרט שעולה $O(n)$ צעדים.
- <u>שלב 3-4</u>: עוברים על תוכן הסרט כדי לוודא שהכל נמחק - עולה לנו $O(n)$ צעדים.

סך הכל, זמן הריצה של המכונה M_2 הוא במקרה הגרוע ביותר $O(2n\log(n) + 2n) \approx O(n\log(n))$.

<u>מסקנה</u>: $L_1 \in DTIME(n \cdot \log(n))$ וגם $L_1 \in DTIME(n^2)$.

<u>תרגיל</u>: האם כל שפה ששייכת ל-$DTIME(n \cdot \log(n))$ שייכת גם ל-$DTIME(n^2)$?.

<u>תשובה</u>: כן, אפשר לייצר צעדים מיותרים ב-$n \cdot \log(n)$ כדי להפוך אותו ל-n^2.

משפטים

משפט 1: אם $t_1(n) = O(t_2(n))$ אז $DTIME(t_1(n)) \subseteq DTIME(t_2(n))$.

חסמים אסימפטוטיים:

- O - חסום מלמעלה. למשל $n = O(n \log n)$ אז נאמר ש: "n חסום ב-$n \log n$".
- Ω - חסום מלמטה.
- Θ - חסום הדוק. למשל $\binom{n}{k} = \Theta(n^k)$, כלומר אפשר לחסום גם מלמעלה וגם מלמטה.
- o - קטן ממש. למשל $t_1(n) = o(n^2)$ אז נאמר ש: "t_1 קטן ממש מ-n^2" ונסמן $t_1 \ll n^2$.
- ω - גדול ממש. למשל $t_1(n) = \omega(n^2)$ אז נאמר ש: "t_1 גדול ממש מ-n^2" ונסמן $t_1 \gg n^2$

שאלה: האם אפשר להכריע את $L_1 = \{a^m b^m | m \geq 0\}$ עוד יותר מהר? בזמן $o(n \log n)$?

תשובה: לא במכונה עם סרט אחד! .

משפט 2: שפה שניתנת להכרעה בזמן $O(n \cdot \log(n))$ במכונה עם סרט אחד היא בהכרח רגולרית.

דוגמה 3 - מכונה עם שני סרטים

למה שלא נבנה מכונה עם שני סרטים להכרעת השפה $L_1 = \{a^m b^m | m \geq 0\}$?

תארו מכונה עם שני סרטים להכרעת $L_1 = \{a^m b^m | m \geq 0\}$ והראו שזמן הריצה שלה הוא $O(n)$.

אלגוריתם: המכונה M_3 על הקלט w תבצע:

1. עבור על סרט הקלט (סרט 1):
   a. סרוק עד ה-b הראשון וממנו העתק את כל ה-b-ים לסרט השני.
   b. אם במהלך ההעתקה נפגשת ב-a לאחר ה-b-ים - אז דחה.
2. עבור במקביל על סרט 1 וסרט 2:
   a. אם מספר התא של ה-$\bar{b}$ הראשון של הסרט הראשון הוא גם מספר התא של ה-$\bar{b}$ הראשון של הסרט השני - אז קבל.
   b. אחרת - דחה.

זמן ריצה: מבצעים מעברים ליניאריים על הסרטים ולכן זמן הריצה של M_3 הוא $O(n)$.

במכונה חד-סרטית הזמן הטוב ביותר הוא $O(n \cdot \log(n))$.

הבדל בין חישוביות וסיבוכיות

בחישוביות: כל המודלים הסבירים שקולים ולכן כל מה שאפשרי לביצוע באחד המודלים אפשרי לביצוע גם בכל המודלים האחרים.

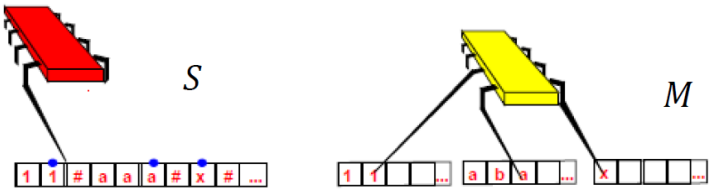
בסיבוכיות: גם אם המודלים M_1, M_2 שקולים, (בה"כ) בחירת M_1 משפיעה על זמן הריצה מבחירת M_2.

כלומר, בסיבוכיות לבחירת המודל יש השפעה על זמן הריצה.

בעיה: תלות במודל

בעיה: באיזה מודל נבחר לצורך סיווג שפות לפי זמן הריצה של המכונות המכריעות אותן?
תשובה: למזלנו, הזמן הדרוש אינו שונה באופן מהותי במודלים דטרמיניסטיים שונים.
נזכר כיצד הוכחנו שמכונה עם כמה סרטים שקולים בכוח החישוב שלה למכונה עם סרט אחד.

תזכורת: הערה על זמן הריצה



כדי לחקות צעד אחד של פעולת המכונה M (רב סרטית), המכונה S (החד-סרטית) עוברת על הסרט שלה מספר פעמים שחסום על ידי קבוע.
אם מספר הצעדים של M הוא $t(n)$, אז מספר התאים המקסימלי שעליהם הראש של S עובר במעבר אחד על הסרט הוא $O(t(n))$.
למה? - כי הוכחנו בתחילת הקורס עבור שקילות רב-סרטי, שהמכונה החד-סרטית תעתיק לימין כל סרט מהרב סרטי לסרט היחיד שלה, ועבור כל העתקה חדשה היא תכתוב בסימן מיוחד ש: "כאן מתחיל סרט כלשהו"..
כלומר המכונה החד-סרטית תכיל "שרשור" של סרטים מהרב סרטי. לכן, מספר הצעדים של M הוא יהיה מספר התאים המקסימלי של הסרט של S.

מסקנה: זמן הסימולציה של מכונה מרובת סרטית ע"י מכונה חד-סרטית הוא **ריבועי לכל היותר**: $O(t(n)^2)$.
למה?: כי לכל סרט i ב-M, צריך לעבור ב-S על כל ה-i $1 \leq j \leq$ סרטים כדי לגשת אליו.

הפער הריבועי הוא הדוק

כמו שראינו, הפער בין מכונה מרובת סרטים למכונה עם סרט אחד הוא **לכל היותר ריבועי**.
למעשה, יש שפות **שהוכח עליהן** שזהו הפער.
למשל, שפת הפלינדרומים מעל $\{a, b\}$:

1. ניתנת להכרעה במכונה דו-סרטית בזמן שהוא $O(n)$.
2. ניתנת להכרעה במכונה חד-סרטית בזמן שהוא $O(n^2)$.
3. לא ניתנת להכרעה במכונה חד-סרטית בזמן שהוא $o(n^2)$ - כלומר "או-קטן".

מודלים לא דטרמיניסטיים

הראינו שזמן הריצה לא שונה באופן מהותי במודלים דטרמיניסטיים שונים.
מצד שני, בחירת מודלים אי-דטרמיניסטיים היא בחירה קריטית שיכולה להשפיע רבות על זמן הריצה.
תחילה עלינו להגדיר מהו זמן הריצה במקרה זה. - נדבר רק על מכונות מכריעות:

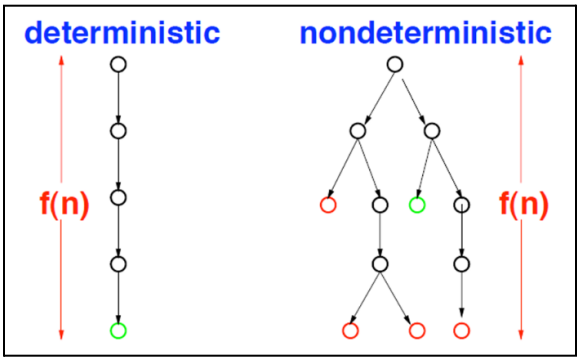
1. כל ענפי החישוב מסתיימים בעצירה (במצב המקבל או הדוחה).
2. נסתכל על זמן החישוב הארוך ביותר.

הגדרה: זמן ריצה לא דטרמיניסטי

תהי N מ"ט א"ד ותהי פונקציה $t: \mathbb{N} \to \mathbb{N}$ המתארת את מספר הצעדים של המכונה.

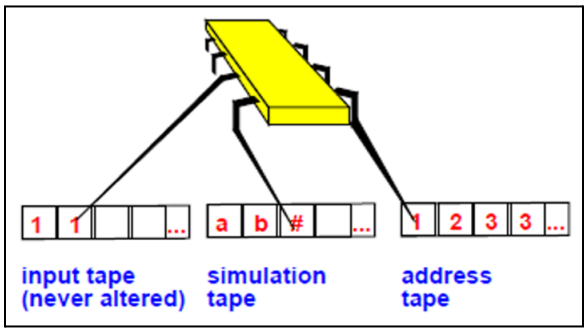
נאמר ש**זמן הריצה** של N הוא $t(n)$, אם לכל קלט w באורך n, מספר צעדי הריצה של N על w הוא לכל היותר $t(n)$, **בכל אחד ממסלולי החישוב** של N על w.

<u>הערה</u>: מסתכלים על זמן הריצה של <u>כל מסלולי החישוב</u>, גם של אלה שמסתיימים **בדחייה**.



דטרמיניסטי ולא דטרמיניסטי

נשים לב שבמסלולי חישוב <u>לא מקבלים</u>, המכונה הלא דטרמיניסטית חייבת להגיע ל-q_{reject} בתוך מגבלת הזמן.

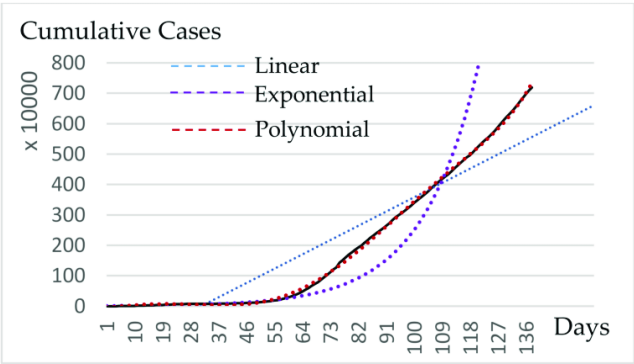


תזכורת לסימולציה

המכונה הדטרמיניסטית D שמסמלצת את המכונה הלא דטרמיניסטית N. המכונה D שתיארנו בחישוביות לא יעילה מבחינת זמן הריצה כי יש בה חזרות על חלקים של חישובים. אפשר לייעל את פעולתה של D, אך עדיין זמן הריצה של כל סימולציה אפשרית של מכונה לא דטרמיניסטית על ידי מכונה דטרמיניסטית הוא **לפחות מעריכי (אקספוננציאלי)** בזמן הריצה של המכונה הלא דטרמיניסטית. כי אם אנחנו סורקים בצורה דטרמיניסטית את המכונה האי דטרמיניסטית אז אנחנו בעץ ולכן זו סריקה בזמן אקספוננציאלי.

- זמן הריצה של מכונה א"ד מכריעה N על מילה w מוגדר כמספר הצעדים במסלול החישוב הארוך ביותר של N על w (מקבל או דוחה).

אבחנה חשובה



- יש לכל היותר פער **פולימוניאלי** בין מכונה מרובת סרטים למכונה עם סרט אחד.
- יש לכל היותר פער **אקספוננציאלי** בין מכונה א"ד למכונה דטרמיניסטית.
- יש הבדל עצום בין הגידול של זמני ריצה **פולינומיאליים** טיפוסיים כמו n^3 לבין הגידול של זמני ריצה **אקספוננציאליים** טיפוסיים כמו 2^n.

אקספוננציאלי לעומת פולינומיאלי

- זמן ריצה **אקספוננציאלי** מצביע, על סריקה ממצה של כל מרחב הפתרונות (למשל חיפוש שלם).
- זמן ריצה **פולינומיאלי** מצביע על הבנה מעמיקה יותר של הבעיה.
- כל המודלים החישוביים הדטרמיניסטיים "הסבירים" **שקולים פולינומיאלית**.

כל אחד מהם יכול לסמלץ כל אחד אחר בהפסק זמן פולינומי. הפסק הזמן הוא לא יותר מאשר ריבועי.

	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.00001 second	.00004 second	.00009 second	.00016 second	.00025 second	.00036 second
n^3	.00001 second	.00008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minute	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	$2 \cdot 10^8$ centuries	$1.3 \cdot 10^{13}$ centuries

השגת אי-תלות במודל

כדי שלא נצטרך לבדוק מה המכונה יכולה ומה לא, אז אנחנו נבחין בין פולינומי לאקספוננציאלי.

התשובה לשאלה "האם זמן הריצה הוא ליניארי, ריבועי וכדומה?" **תלויה במודל**.

התשובה לשאלה "האם זמן הריצה פולינומיאלי?" **לא תלויה במודל**.

בסופו של דבר אנחנו מעוניינים במדד לקושי של **חישובים** (מבחינת זמן הריצה) ולא בהתמקדות במודל כזה או אחר. לכן נבחין בין זמן ריצה פולינומי לזמן ריצה **על-פולינומי** (אקספוננציאלי). במילים אחרות, לא יעניין אותנו מה כתוב במכונה ומה היא עושה, השאלה שנשאל זה האם היא פולינומית או על פולינומית.

הגדרה: מכונת טיורינג פולינומית

- מ"ט דטרמיניסטית M תקרא **פולינומית**, אם קיים פולינום $p(n) = n^c$ עבור $c \in \mathbb{R}$ כך שלכל מילה $x \in \Sigma^*$, המכונה M עוצרת על x בתוך לכל היותר $p(|x|)$ צעדים.
- <u>במילים</u>: מ"ט דטרמיניסטית M תקרא **פולינומית**, אם יש איזשהו פולינום $p(n)$ שחוסם את זמן הריצה של M לכל קלט $x \in \Sigma^*$.
- מ"ט א"ד M תקרא **פולינומית**, אם קיים פולינום $p(n) = n^c$ עבור $c \in \mathbb{R}$ כך שלכל מילה $x \in \Sigma^*$, המכונה M עוצרת על x בתוך לכל היותר $p(|x|)$ צעדים, **בכל מסלולי החישוב שלה**.
- הפולינום $p(n) = n^c$ יקרא **חסם זמן הריצה** של M.

מבנה הוכחת נכונות של מכונה פולינומית

מכונה דטרמיניסטית פולינומית מכריעה:

1. <u>הוכחת נכונות</u>:
   a. $M(x) = 1 \leftarrow x \in L$
   b. $M(x) = 0 \leftarrow x \notin L$
2. <u>הוכחת סיבוכיות</u>: להוכיח שלכל קלט x, זמן הריצה של המכונה על x חסום ע"י פולינום כלשהו $p(|x|)$.

מכונה א"ד פולינומית מכריעה:

1. <u>הוכחת נכונות</u>:
   a. $x \in L \leftarrow$ אז **קיים** מסלול **מקבל** בעץ החישוב של המכונה על x.
   b. $x \notin L \leftarrow$ אז **כל** המסלולים בעץ החישוב של המכונה על x **דוחים**.
2. <u>הוכחת סיבוכיות</u>: להוכיח שלכל קלט x, **ולכל מסלול בעץ החישוב**, זמן הריצה של המכונה על x חסום ע"י פולינום כלשהו $p(|x|)$.

המחלקות P ו-NP

הגדרה: המחלקה P

הגדרה: המחלקה P היא מחלקת השפות כך שלכל שפה:

1. יש מכונת טיורינג דטרמיניסטית מכריעה.
2. זמן הריצה של המכונה הוא פולינומיאלי (בגודל הקלט).

$$P = \cup_{k\geq 0} DTIME(n^k)$$

הסבר: $L \in P$ אם יש אלגוריתם בעל זמן ריצה פולינומיאלי להכרעת השייכות ל-L.

כלומר, קיים אלגוריתם שזמן ריצתו $O(n^k)$ ל-k קבוע טבעי כלשהו.

חשיבות המחלקה P

המחלקה P היא מחלקה חשובה של שפות.

1. שייכות של שפה למחלקה P לא תלויה במודל (במספר הסרטים של המכונה שמכריעה שייכות לשפה).
2. שייכות של שפה למחלקה P לא תלויה במודל החישובי, כל עוד הוא **שקול פולינומיאלי** למ"ט.
3. המחלקה P מתאימה למחלקת הבעיות **הפתירות באופן מעשי** במחשב (כל מה שאנחנו יודעים לפתור, למשל $BFS,\ DFS, Dijkstra$ נמצאות ב-P כי הם פתירות באופן מעשי).

המחלקה מתאימה למציאות

שאלה: האם $O(n^{100})$ גם ב-P?

תשובה: כן. זה ייחשב בזמן ריצה פולינומיאלי. אבל במציאות זמן הריצה של אלגוריתמים הוא או פולינומיאלי עם **פולינום צנוע** או על-פולינומיאלי (אקספוננציאלי).

הערה: הניסיון מראה שהסף שאותו צריך לעבור הוא המעבר מזמן ריצה אקספוננציאלי לפולינומיאלי. ברגע שהמעבר הזה קורה, נמצאים אלגוריתמים יותר ויותר יעילים.

דוגמאות לבעיות ב-P

1. **אריתמטיקה**: חיבור, חיסור, כפל, חילוק עם שארית וכו'...
2. **מספרים**: מציאת מחלק משותף מקסימלי (GCD).
3. **חקר ביצועים**: זרימה ברשתות, תכנון ליניארי.
4. **אלגברה**: כפל מטריצות, דטרמיננטות וכו'...
5. **גרפים**: $DFS,\ BFS$, מציאת עץ פורש מינימלי (קרוסקל), מעגל אוילר ועוד...

ניתוח טיפוסי של זמן ריצה

1. מחלקים את האלגוריתם לשלבים.
2. מראים שזמן הריצה של כל שלב פולינומיאלי.
3. מראים שמספר הפעמים שכל שלב מתבצע הוא פולינומיאלי.
4. מסיקים שזמן הריצה של האלגוריתם כולו פולינומיאלי:
   a. **סכום** של פולינומים הוא פולינום.
   b. **כפל** של פולינומים הוא פולינום.
   c. **הרכבת** פולינומים הוא פולינום.

קידוד הקלט

זמן הריצה מוגדר כפונקציה של גודל הקלט (מספר הביטים). לכן, **צריך לבחור קידוד סביר לקלט** ולא קידוד שיגדיל באופן מלאכותי את גודל הקלט. (למרות שמלאכותי זמן הריצה ייראה טוב יותר - זה נקרא "ריפוד").

<u>דוגמה - קידוד של גרף</u>: רשימה של צמתים ורשימה של קשתות, או מטריצת שכנויות.

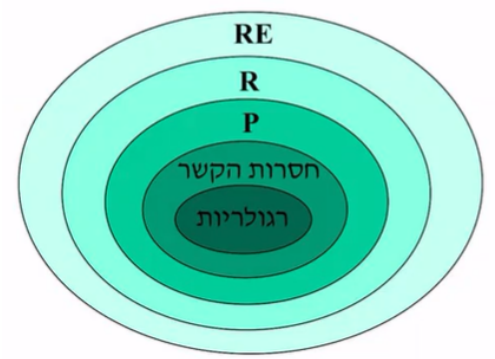
1. אפשר לעבור מקידוד מסוים לקידוד אחר בזמן פולינומיאלי, והגודל של כל קידוד פולינומיאלי בגודל של הקידוד האחר.
2. אנחנו רגילים לחשב את זמן הריצה כפונקציה של **מספר הצמתים**. זה לא גודל הקלט, אבל גודל הקלט פולינומיאלי במספר הזה. למשל גרף הנתון כמטריצת שכנויות: גודל הקלט: $O(|V|^2)$.

<u>דוגמה</u>: **מספרים** יכולים להיות מיוצגים בבינארי או בעשרוני, **אבל לא באונרי**.
למה לא אונרי? - כי הגודל של הקידוד האונרי אקספוננציאלי בגודל של הקידוד הבינארי (או העשרוני).

<u>הערה חשובה</u>: אלגוריתם על מספרים ייחשב פולינומיאלי אם זמן הריצה שלו פולינומיאלי **בכמות הספרות** המייצגות את המספרים (ולא אם הזמן פולינומיאלי במספרים עצמם המיוצגים ע"י הספרות).

1. פולינומיאליות בגודל הקלט.
2. <u>לדוגמה</u>: אם $n = 1000000$ (בייצוג בינארי), אז זמן הריצה צריך להיות פולינומיאלי ב-7.

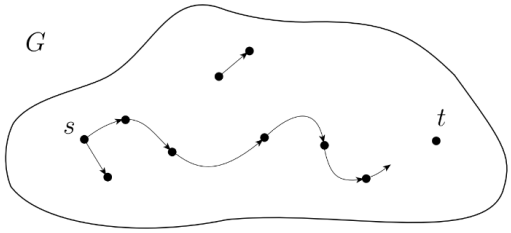
תמונת עולם של המחלקות



כל ההכלות בין המחלקות הן הכלות ממש.
למה P בתוך R? - כי אמרנו ש-P זה מחלקה של שפות כך שלכל שפה קיימת מ"ט דטרמיניסטית פולינומית <u>המכריעה</u> אותה. לכן, כל P נמצא ב-R. נשים לב ש: $P \subset R$ כלומר $P \neq R$.

דוגמה עם הסבר (לא מההרצאה)

בעיית ה-$PATH$ נועדה כדי להכריע האם קיים מסלול מכוון בגרף בין הצמתים s ל-t.



נתונה השפה:

$$PATH = \{< G, s, t, > \mid G\ is\ a\ directed\ graph\ that\ has\ a\ directed\ path\ from\ s\ to\ t\}$$

<u>טענה</u>: $PATH \in P$.

<u>רעיון ההוכחה</u>: נוכיח את הטענה ע"י הצגת אלגוריתם (מכונה) מכריע לשפה $PATH$ בזמן ריצה פולינומי.
לפני שנתאר את האלגוריתם, נראה למה אלגוריתם $Brute\ Force$ לא מהיר מספיק עבור הבעיה הזאת.
אלגוריתם $Brute\ Force$ עבור $PATH$ בוחן את כל המסלולים הפוטנציאלים ב-G וקובע האם אחד מהם יש מסלול מכוון בין s ל-t. מסלול פוטנציאלי הוא סדרה של צמתים ב-G באורך של לכל היותר $m = |V(G)|$.
אבל מספר המסלולים הפוטנציאלים האלה הם לכל היותר m^m, שזה אקספוננציאלי במספר הצמתים ב-G.
לכן, אלגוריתם $Brute\ Force$ משתמש בזמן ריצה אקספוננציאלי - לא רלוונטי לפתרון שלנו.
כדי ליצור אלגוריתם (מכונה) מכריע עבור $PATH$ בזמן ריצה פולינומי, אנחנו חייבים למנוע ממנו להיות $Brute\ Force$. אחת הדרכים למנוע דבר כזה הוא באמצעות שיטה לחיפוש בגרף בשם BFS. כאן, אנחנו נסמן את כל הצמתים לאורך המסלול המכוון החל מ-s. ככה אפשר יהיה לחסום את זמן הריצה לפולינומיות.

<u>הוכחה</u>: האלגוריתם M על $PATH$ עובד בזמן פולינומי לפי התיאור הבא:
המכונה M על הקלט $< G, s, t >$ מבצעת:

1. סמן את הצומת ההתחלתית s.
2. חזור על השלבים הבאים עד שלא נותרו צמתים נוספים לסימון:
3. עבור על כל הצלעות של G. אם צומת (a, b) נמצאה שעוברת מהצומת המסומנת a לצומת הלא-מסומנת b, אז סמן גם את b.
4. אם t סומנה, <u>קבל</u>. אחרת, <u>דחה</u>.

כעת נתחם את האלגוריתם כדי להראות שהוא רץ בזמן פולינומי. שלבים 1 ו-4 פועלים רק פעם אחת. שלב 3 רץ לכל היותר $|V(G)|$ פעמים כי בכל צעד חוץ מהאחרון, נסמן צומת נוספת. לכן, מספר הצעדים הוא לכל היותר $|V(G)| + 1 + 1$, שזה פולינומיאלי בגודל של G. שלב 3 מבצע חיפוש בהתאם לקלט ובוחן האם צמתים מסוימים סומנו, לכן זה ניתן למימוש בזמן פולינומי.
ומכאן M הוא אלגוריתם בזמן פולינומי המכריע את השפה $PATH$, כלומר $PATH \in P$.

תכונות סגירות של P

<u>תרגיל</u>: הוכיחו שהמחלקה P סגורה ל:

1. **איחוד**
2. **חיתוך**
3. **משלים**
4. **שרשור** - אם יש 2 בעיות שצריך לפתור ואנחנו משרשרים אותם, אז יש לנו אלגוריתם פולינומים לבעיה הראשונה, נריץ אותה: אם תקין, אז נעבור לאלגוריתם הפולינומי של המכונה השניה ונריץ גם אותו: אם תקין אז גם נשארנו פולינומי ולכן נשארנו ב-P.
5. **איטרציה** - בעזרת אלגוריתם מסוג תכנות דינאמי (לא צריך להוכיח את זה בקורס).

<u>הערה</u>: $P \subset R$ כי, $P \neq R$.

המחלקה NP

<u>תזכורת - מחלקת השפות לדטרמיניסטי</u>: תהי פונקציה $t: \mathbb{N} \to \mathbb{N}$. נסמן ב-$DTIME(t(n))$ את <u>מחלקת השפות</u> שיש להן מכונת טיורינג <u>דטרמיניסטית מכריעה</u> שזמן הריצה שלה הוא $O(t(n))$.
פורמלית: $DTIME(t(n)) = \{L \mid L\ is\ a\ language,\ decided\ by\ an\ O(t(n)) - time\ DTM\}$.

<u>הגדרה - מחלקת השפות לא"ד</u>: תהי פונקציה $t: \mathbb{N} \to \mathbb{N}$. נסמן ב-$NTIME(t(n))$ את <u>מחלקת השפות</u> שיש להן מכונת טיורינג <u>לא דטרמיניסטית מכריעה</u> שזמן הריצה שלה הוא $O(t(n))$.
פורמלית: $NTIME(t(n)) = \{L \mid L\ is\ a\ language,\ decided\ by\ an\ O(t(n)) - time\ NTM\}$.

<u>תזכורת</u>: זמן הריצה של מכונה א"ד מוגדר לפי מסלול החישוב <u>הארוך ביותר</u> על המילה.

<u>הגדרה</u>: המחלקה NP היא מחלקת השפות כך שלכל שפה:

1. יש מכונת טיורינג **לא-דטרמיניסטית מכריעה**.
2. זמן הריצה של המכונה הוא **פולינומיאלי** (בגודל הקלט).

$$NP = \cup_{k \geq 0} NTIME(n^k)$$

הסבר לא פורמלי של NP - "עד" ומוודא - Validator

השפה $L \in NP$ אם לכל מילה $w \in L$ יש דרך **לוודא במהירות** ש-w שייכת לשפה הזאת.

1. "במהירות" == בזמן פולינומיאלי בגודל הקלט.
2. לכל מילה בשפה יש "**תעודת שייכות**" שמשכנעת שהמילה שייכת לשפה.
3. לכל מילה <u>שלא שייכת לשפה</u> אין תעודת שייכות כזאת.
4. לפעמים נקרא לתעודה זו **"עד"** או באנגלית: witness.

תעודת שייכות

אמרנו שלכל מילה בשפה יש "**תעודת שייכות**" שמשכנעת שהמילה שייכת לשפה.
למשל, באלגוריתם למציאת מעגל המילטון בגרף, נרצה לנחש את פרמוטציית הסידור הנכונה מתוך כל ה-$!|V|$ פרמוטציות, שתגיד לנו "קיים מעגל המילטון בגרף".

- אם המילה בשפה, אז הסידור שניחשנו הוא **"תעודת השייכות"** שלנו שמשכנעת ש: $< G > \in L$.
- אם המילה לא בשפה, אז לא משנה איזה **"עד"** ניתן, לא קיימת אף "תעודת שייכות" שתשתכנע אותנו שאכן קיים מעגל המילטון בגרף ומכאן $< G > \notin L$. כלומר לא משנה איזה סידור ננחש, לעולם לא קיים מעגל המילטון בגרף.

תכונות של המחלקות P,NP

תכונה 1: $R \subseteq P$ וגם $R \subseteq NP$.
הסבר 1: מכיוון ששתי ההגדרות של P ו-NP מתייחסות למ"ט המכריעות שפות, אז הם מוכלות ב-R.

תכונה 2: $P \subseteq NP$.
רעיון הוכחת תכונה 2: הוכחנו שהמודל הדטרמיניסטי שקול למודל הלא דטרמיניסטי. ובפרט, ראינו שקל להפוך כל מכונה דטרמיניסטית למכונה לא דטרמיניסטית. אבל, כעת צריך לשים לב **לחסם הפולינומי** לזמן הריצה של המכונה.
הוכחת תכונה 2: תהי $L \in P$ ותהי M_L מ"ט דטרמיניסטית פולינומית המכריעה את L.
יהי $p(n)$ חסם בזמן הריצה הפולינומי של M_L.

1. נתאר מכונה א"ד פולינומית M_L' המכריעה את L.
2. המכונה M_L' תהיה זהה ל-M_L למעט שינוי פורמלי בהגדרת פונקציית המעברים של המכונה M_L (כדי שתתאים למודל האי"ד).
3. נשים לב שלכל קלט $x \in \Sigma^*$ עומק העץ חסום ע"י החסם של המכונה המקורית, $p(|x|)$.

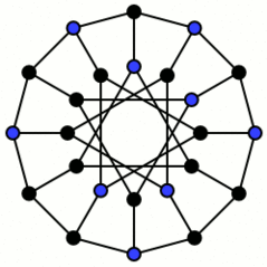
נכונות הבניה:

- $x \in L \rightarrow M_L(x) = 1 \rightarrow$ **קיים** מסלול בעץ $\tilde{\delta}$ כך ש: $M_L'(x) = 1$.
- $x \notin L \rightarrow M_L(x) = 0 \rightarrow$ **לכל** מסלול בעץ $\tilde{\delta}$ כך ש: $M_L'(x) = 0$.

האם $NP \subseteq P$?

במילים אחרות - האם אפשר להמיר מ"ט א"ד פולינומית למ"ט דטרמיניסטית פולינומית?.
תשובה: לא ידוע. כשהוכחנו בשיעור הקודם (של מ"ט א"ד) ראינו שניתן לסמלץ מ"ט א"ד ע"י מ"ט דטרמיניסטית, התעלמנו מכמות המשאבים הדרושים לכך. לכן, אם נרצה לסרוק את כל עץ החישוב של המכונה הלא דטרמיניסטית הפולינומית, נקבל זמן ריצה פרופורציונלי לגודל העץ, כלומר $\Omega(2^{p(n)})$ - וזה אקספוננציאלי!. השאלה הזו נקראת שאלת **P=NP?** זו שאלה פתוחה במדמ"ח שלא הצליחו להוכיח/להפריך.

דוגמא לשפה ב-NP (השפה IS)



<u>הגדרה</u>: בתורת הגרפים, **קבוצה בלתי תלויה** (IS - Independent set) היא קבוצת קודקודים בגרף, אשר אין זוג מביניהם המחוברים ישירות דרך קשת אחת. למשל בציור משמאל: כל הצמתים הכחולים נמצאים בקבוצה IS כי בין כל צומת כחול אין שכן שהוא גם כחול.

נתונה השפה:

$$IS = \{< G, k > \mid G\ is\ a\ graph\ that\ contains\ an\ independent\ set\ of\ size\ k\}.$$

<u>טענה 1</u>: $IS \in R$.

<u>רעיון ההוכחה 1 - אלגוריתם חיפוש שלם</u> -

1. אלגוריתם של חיפוש מלא יכול לעבור על כל ה- $\binom{n}{k}$ אפשרויות לבחירת k קודקודים מהגרף.
2. לכל אפשרות כזו, לבדוק האם היא קבוצה ב"ת (לבדוק שכל זוג צמתים בקבוצה לא מחוברים בצלע).
3. אם מצאנו אפשרות שכזו - <u>קבל</u>.
4. אם עברנו על כל האפשרויות ולא מצאנו - <u>דחה</u>.

<u>הערה 1</u>: יש לנו $\binom{n}{k}$ תתי קבוצות בגודל k, איזה אחד מהם אנחנו צריכים? - אנחנו צריכים לעבור על כולם ולכן צריך לעבור על $2^{\binom{n}{k}}$ בחיפוש שלם.

<u>טענה 2</u>: $IS \in NP$.

<u>רעיון ההוכחה 2</u>: במקום חיפוש מלא, ניתן **לנחש** קבוצה המועמדת להיות קבוצה ב"ת בגרף ולבדוק רק אותה.

<u>הערה 2</u>: אם שפה ב-NP אז יש לנו "תעודת שייכות" שיגיד "אני ב-NP", במקרה שלנו מהי התעודה?: "קבוצה של k קודקודים". מ"ט תריץ מסלול פשוט: **1.** לעבור על ה-k צמתים מהקבוצה. **2.** לכל צומת לוודא שאין שכנים בקבוצה. לכל זמן הריצה במקרה זה יהיה $O(k^2)$. אם כך, $k \leq |V(G)|$ וגם $|G| = O(|V(G)^2|)$.

<u>הוכחה פורמלית של טענה 2</u>: נתאר מ"ט א"ד פולינומית המכריעה את השפה IS. נוכיח ש: $IS \in NP$.

<u>אלגוריתם</u>: N על קלט $< G, k >$ תבצע:

1. נחש תת קבוצה S של צמתי G. (בדוק שגודל הקבוצה הינו k).
2. לכל זוג צמתים מהקבוצה $v_1, v_2 \in S$, בדוק שאכן $\{v_1, v_2\} \notin E(G)$.
   a. אם לאחת הצלעות גילינו ש: $\{v_1, v_2\} \in E(G)$ - אז <u>דחה</u>.
3. אם עברנו על כל הזוגות ולא דחינו - אז <u>קבל</u>.

<u>נכונות האלגוריתם</u>:

- אם $< G, k > \in IS$ אזי **קיימת** תת-קבוצה $S \subseteq V$ בגודל k שהיא ב"ת. ← קיים מסלול חישוב שבו המכונה N מנחשת בדיוק את הקבוצה S ← במסלול זה, הבדיקה של N על הקבוצה S יוצאת תקינה ולכן N מקבלת. ← קיים מסלול מקבל. ← $< G, k > \in L(N)$ כנדרש.

- אם $< G, k > \notin IS$ אזי **לכל** תת-קבוצה $S \subseteq V$ בגודל k שהיא **תלויה**. ← **לכל** ניחוש של קבוצה S, המכונה N תזהה שקיימת צלע ב-S ותדחה ← כל המסלולים **דוחים** $< G, k > \notin L(N)$ ← כנדרש.

הוכחנו כי $IS = L(N)$ ולכן המכונה N אכן מכריעה את השפה IS.

סיבוכיות האלגוריתם: הסיבוכיות של ניחוש הוא כגודל הניחוש (במקרה זה $O(k)$).

- גודל הקלט: $O(n^2)$ - (מטריצת שכנויות) כאשר n הינו מספר הצמתים בגרף הנתון.
- נניח כי $k \leq n$.
- בשלב 1: ניחוש קבוצה בגודל k עולה לנו $O(k)$.
- בשלב 2: עוברים על $O\left(\binom{k}{2}\right) = O(k^2)$ זוגות של צמתים.
- סך הכל: $O(k^2 + k) = O(k^2)$ - **זמן ריצה פולינומי**.

ניסוח שקול לשאלת P=NP

שאלה: האם וידוא יעיל גורר חיפוש יעיל?

תשובה: וידוא יעיל הוא מאפיין מרכזי של NP. (כלומר, בהינתן הצעה לפתרון אפשרי, ניתן בזמן פולינומי לוודא שהפתרון המוצע נכון).

דוגמה: משחק סודוקו, בהינתן פתרון של סודוקו, אפשר בקלות יחסית (=באופן פולינומי) לוודא שהפתרון נכון.

השאלה היא: אם התכונה הנ"ל גם אומרת **שמציאת** הפתרון יכולה להתבצע ביעילות?

דוגמה לבעיה כריעה שאיננה ב-NP: משחק שחמט. אם במהלך המשחק יציגו לנו צעד שאותו יש לבצע כדי לנצח, אין לנו אפשרות לדעת בקלות שהצעד המדובר באמת יביא לנו ניצחון. (כלומר גם **בדיקה** של פתרון אופציונלי דורשת הרבה עבודה).

המחלקה coNP

מחלקת השפות $coNP$: השפות שהמשלימה שלהן שייכת ל-NP. כלומר $coNP = \{L | \bar{L} \in NP\}$.

באופן לא פורמלי, שפה L שייכת למחלקה $coNP$ אם לכל מילה w **שלא שייכת** ל-L יש דרך מהירה לוודא ש-w **לא שייכת** ל-L.

תרגיל: אלו שפות נראות קשות יותר, אלה שב-NP או אלה שב-$coNP$?

הקבלה לשפות מתורת החישוביות

המחלקות $P,\ NP,\ coNP$ מזכירות מחלקות מתורת החישוביות.

- המחלקה P מקבילה למחלקת השפות ב- R. (כי יש לשפה מכונה מכריעה).
- המחלקה NP מקבילה למחלקת השפות ב-RE. (כי יש לשפה מאמת).
- המחלקה $coNP$ מקבילה למחלקת השפות ב-$coRE$.

משפט: $P \subseteq coNP$ (**ההוכחה בהמשך הקובץ - שאלה 11 - תרגול 5**).

יחס R_L

<u>הגדרה</u>: שפה $L \in NP$ אם קיים **יחס דו-מקומי** R_L המכיל זוגות (x, y) כך שמתקיים:

1. $x \in L$.
2. y הינו **"עד"** עבור x. (כלומר, y הוא אובייקט המעיד על השייכות של x לשפה L).

<u>היחס R_L מקיים</u>:

3. R_L חסום פולינומית.

   כלומר: לכל זוג $(x, y) \in R_L$ קיים פולינום $p(\cdot)$ כך שמתקיים: $|y| \leq p(|x|)$.

4. היחס R_L ניתן לזיהוי דטרמיניסטי פולינומי. כלומר - קיימת מ"ט דטרמיניסטית פולינומית V המכריעה את היחס (לכל זוג (x, y) תענה האם $(x, y) \in R_L$ או $(x, y) \notin R_L$). המכונה V נקרא **מאמת**.

5. השפה L מקיימת: $L = \{x \in \Sigma^* \mid \exists y \; s.t. \; (x, y) \in R_L\}$.

כלומר לכל מילה בשפה **קיים** עד פולינומי, ולכל מילה שלא בשפה **לא קיים** עד פולינומי.

- $x \in L \leftarrow$ קיים פולינום $p(\cdot)$ וקיים $y \in \Sigma^{p(|x|)}$ כך ש: $(x, y) \in R_L$.
- $x \notin L \leftarrow$ לכל פולינום $p(\cdot)$ ולכל $y \in \Sigma^{p(|x|)}$ כך ש: $(x, y) \notin R_L$.

<u>במילים</u>: נשאל האם הבעיה x נפתרת בעזרת העד y, במידה וכן אז $(x, y) \in R_L$ ואחרת לא ביחס.

<u>הגדרה שקולה</u>: $L \in NP$ אם קיימת V מכונה <u>מוודאות</u> דטרמיניסטית מכריעה ופולינומית עבור L.

קיים פולינום $p(\cdot)$ וגם עד y עבור V ו-x כל שלכל $x \in \Sigma^*$ מתקיים:

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{p(|x|)} : V(x, y) = 1$$
$$x \notin L \Leftrightarrow \forall y \in \{0,1\}^{p(|x|)} : V(x, y) = 0$$

<u>הסבר</u>: מכונה מוודאת V מקבלת את הקלט $x \in \Sigma^*$ וגם מקבלת באופן **חיצוני** את y שנקרא לו **העד**.
העד y עוזר לנו להגיד האם $x \in L$ או לא, כלומר, הוא יתן לנו עדות על השייכות ליחס של השפה.
ה-y הוא בעצם איזשהו פרמוטציה של מחרוזת בינארית שנקבל באופן חיצוני לא ידוע.

למשל עבור השפה $HC = \{< G > \mid G \; has \; a \; Hamiltonian \; cycle\}$. בהינתן $x \in \Sigma^*$ כלשהו ובהינתן עד y שהגיע באופן חיצוני. המוודא $V(x, y) = 1$ אם העד y מייצג מעגל המילטון (פרמוטציה/סידור של צמתי הגרף) וגם $x =< G >$. כלומר $V(< G >, Hamiltonian \; cycle) = 1$ שאכן במקרה זה $< G > \in HC$.
חשוב להבין שלא תמיד נקבל y שהוא בהכרח מעגל המילטון, יכול להיות שנקבל y שידחה. כלומר, העד לא תמיד מבטיח לנו שהמילה תתקבל בשפה, והאחראי להחלטה הזאת היא המכונה המוודא V.

הוכחת $IS \in NP$ על ידי ההגדרה השנייה

"עד" עבור השפה IS יהיה קבוצה ב"ת בגודל k.

היחס R_{IS} מכיל זוגות מהצורה $(< G,k > \ , \ < S >)$.

גודל העד הוא $O(k)$ - שזה פולינומי בגודל הגרף G.

<u>לפי התנאי השלישי</u>:

- אם $< G,k > \in IS$ ← קיימת תת-קבוצה $S \subseteq V$ בגודל k שהיא ב"ת ← נבחר אותה להיות "העד".
- אם $< G,k > \notin IS$ ← **לכל** תת-קבוצה $S \subseteq V$ בגודל k, מתקיים ש-S היא קבוצה תלויה. ← לכל תת קבוצה שנבחר להיות "העד", לא יתקבל זוג אשר שייך ליחס R_L.

<u>תרגיל</u>: כתבו פסאודו-קוד של פעולת המאמת V.

טענה: שתי ההגדרות של NP הן שקולות

<u>הוכחה</u>: ראשית נסמן:

- NP_1 = המחלקה NP לפי ההגדרה ה-1: כלומר כל השפות שיש להן מ"ט א"ד פולינומית מכריעה.
- NP_2 = המחלקה NP לפי ההגדרה ה-2 המשתמשת ביחס: כלומר כל השפות כך שקיים עבורן יחס R_L שהוא חסום פולינומית, ניתן לזיהוי יעיל, וכן לעל מילה בשפה **קיים** עד פולינומי, ולכל מילה שאיננה בשפה **לא קיים** עד פולינומי.

<u>כיוון א'</u>: $NP_2 \subseteq NP_1$. תהי $L \in NP_2$. תהי V מכונה פולינומית <u>דטרמיניסטית</u> המוודאת את היחס R_L, יהי $p_1(n)$ חסם פולינומי לזמן הריצה של V ויהי $p_2(n)$ חסם פולינומי לגודל העדים ב-R_L.

<u>האלגוריתם</u>: נתאר מ"ט א"ד N <u>המכריעה</u> את השפה L:

המכונה N על קלט x מבצעת:

1. מנחשת עד y באורך לכל היותר $p_2(|x|)$. (כלומר, העד חסום פולינומית).
2. מסמלצת את ריצת המכונה המוודאת V על הזוג (x,y) ועונה כמוה.

<u>דוגמה לכיוון א'</u>: **1.** היא מנחשת קבוצה של קודקודים **2.** מוודאת שאכן הקבוצה הזאת היא בלתי תלויה.

<u>נכונות האלגוריתם</u>:

- אם $x \in L$ ← קיים "עד" y באורך פולינומי ומתקיים $(x,y) \in R_L$. ← קיים ניחוש נכון של y בעץ החישוב של N על x ← במסלול החישוב הנכון, N <u>מקבלת</u> את x.
- אם $x \notin L$ ← לכל ניחוש של y מתקיים שהזוג $(x,y) \notin R_L$ ← לכל ניחוש y בעץ החישוב של N על x המכונה N <u>תדחה</u> את x.

<u>סיבוכיות האלגוריתם</u>:

- <u>שלב 1</u>: $O(p_2(|x|))$: הניחוש של y זה פולינומי.

- שלב 2: $O(p_1(|x|) + p_2(|x|))$ מוודאת שאכן y הוא "עד", אז היא מריצה את x וגם היא צריכה לקרוא את הניחוש הזה שזה גם יעלה x בזמן פולינומי.
- סך הכל: $O(p_1(|x|) + p_2(|x|))$ פולינומי ב-$|x|$ כנדרש. (כי סכום פולינומים הוא גם פולינום).

לסיום: הראנו שאם יש לנו יחס תקין אז אפשר לכתוב מכונה א"ד.

∎

כיוון ב': $NP_1 \subseteq NP_2$. תהי $L \in NP_1$ ותהי N מ"ט א"ד המכריעה את השפה L ויהי $p(n)$ חסם פולינומי לזמן הריצה של המכונה N (כלומר, לעומק על עצי החישוב שלה).

יחס R_L עבור השפה L היא:

1. לכל $x \in L$ ניקח **מסלול ריצה מקבל** בעץ החישוב של N על x להיות עד. נייצג כל מסלול ריצה כזה כמחרוזת בינארית (0=שמאלה ו-1=ימינה).
2. האורך של כל מחרוזת כזו חסום ע"י $p(|x|)$.
3. ניתן לבדוק "בקלות" (כלומר, בחישוב דטרמיניסטי בזמן פולינומי) אם המסלול שניתן לנו כ-"עד", מסתיים או לא מסתיים במצב מקבל. (זה בעצם המאמת V).

נכונות:

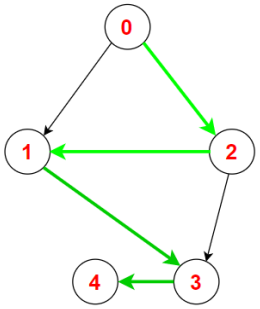
- אם $x \in L$ אז **קיים** מסלול מקבל בעץ החישוב של N על x. מכאן קיים עד y שאורכו חסום ע"י פולינום ב-$|x|$. לכן הזוג (x, y) שייך ליחס R_L.
- אם $x \notin L$ אז כל המסלולים בעץ החישוב של N על x דוחים. מכאן **שלכל** עד y, הזוג $(x, y) \notin R_L$.

לסיום: הראנו שאם יש לנו מכונה א"ד אז אפשר לכתוב לה יחס.

הערה: הראנו שהמסלול הספציפי הוא ה-"עד". ה-"עד" הזה יהיה הקידוד של המסלול.

∎

מסלול המילטון - השפה HAMPATH



מסלול המילטוני בגרף **מכוון** D הוא מסלול פשוט (ללא מעגלים) שמבקר בכל צומת פעם אחת ויחידה.

נתונה השפה הבאה:

$$HAMPATH = \{< D, s, t > \mid D\ is\ a\ directed\ graph\ with\ a\ Hamiltonian\ path\ from\ s\ to\ t\}$$

תרגיל: תארו מאמת בעל זמן ריצה פולינומיאלי לשפה $HAMPATH$.

פתרון: הסבר לפתרון: ה-"עד" יהיה סידור של $n-1$ צמתים והניחוש יהיה "מהו המסלול?".

העד: בהינתן $< D, s, t >$ העד y הינו סידור של שאר $n-2$ הקודקודים.

בנוסף $|y| = O(\sqrt{D})$ כי D זה גרף המיוצג כמטריצת שכנויות ולכן סידור הצמתים יעלה לנו $O(\sqrt{D})$.

המאמת: המאמת V על קלט (x, y):

1. מוודא שכל זוג צמתים סמוכים ב-y מחוברים בצלע. אם לא - דוחה.
2. מוודא צלע s לתחילת y ו-t לסוף y.

זמן ריצה: כדי לוודא שקיימות צלעות כאלה שבונות מסלול יעלה לנו $O(n)$. הסידור עולה $O(\sqrt{D})$.
לכן קיבלנו זמן ריצה פולינומי.

טענה: $HAMPATH \in NP$.

הוכחה: נציג אלגוריתם N על $HAMPATH$ שעובד בזמן פולינומי.

המכונה N על הקלט $< G, s, t >$ מבצעת:

1. כתוב רשימה של $m = |V(G)|$ מספרים, $p_1, \ldots, p_m$. כל מספר ברשימה נבחר באופן לא דטרמיניסטי בטווח $[1, m]$.
2. בדוק אם קיים חזרתיות ברשימה. - אם נמצא כזה, דחה.
3. בדוק האם $s \neq p_1$ וגם $t \neq p_m$. אם מתקיים, דחה.
4. לכל $1 \leq i \leq m-1$, בדוק האם (p_i, p_{i+1}) הוא צלע ב-G. אם לא, דחה.
5. קבל.

כדי לנתח את האלגוריתם ולאמת שהוא רץ בזמן ריצה פולינומי א"ד, נתח כל אחד מהשלבים שלו.
בשלב 1 - קיים בחירה א"ד שמתבצעת בזמן פולינומי. בשלב 2 ו-3, כל שלב רץ בזמן פולינומי. בשלב 4 אנחנו עוברים בלולאה בצורה ליניארית שזה בהחלט לכל היותר פולינומי. לכן N מכונה א"ד מכריעה בזמן פולינומי.
ומכאן $HAMPATH \in NP$.

שאלה: האם $HAMPATH \in P$?

תשובה: לא ידוע על אלגוריתם פולינומיאלי להכרעת השייכות לשפה, וגם לא הוכח שאלגוריתם כזה לא קיים.

השפה COMPOSITES

<u>תזכורת</u>: מספר טבעי נקרא **פריק** אם הוא מכפלה של שני מספרים טבעיים גדולים מ-1.

נתונה השפה: $COMPOSITES = \{< n > | n = pq, for\ integers\ p,q > 1\}$.

<u>שאלה</u>: האם $COMPOSITES \in P$?

<u>תשובה</u>: נשים לב שישנו אלגוריתם דטרמיניסטי שנלמד בעבר שאומר שניתן לעבור על כל המספרים מ-1 ועד $\sqrt{n}$. אם מצאנו מחלקה, אזי הוא פריק ואם לא אז הוא ראשוני. אבל נסתכל מה סיבוכיות האלגוריתם הזה: מאחר שהקלט $< n >$ מיוצג במחרוזת בינארית אז הוא בגודל $\log_2(n)$ ואנחנו נרוץ $\sqrt{n}$ פעמים אז נקבל שזמן הריצה של האלגוריתם הזה יהיה $\sqrt{n} = \sqrt{2^{\log_2(n)}} = 2^{\frac{\log(n)}{2}}$ כלומר זה זמן אקספוננציאלי. במילים אחרות, קיבלנו שהיחס בין גודל הקלט לזמן הריצה הוא אקספוננציאלי $p(\log_2(n)) = 2^{\frac{\log(n)}{2}}$ ולכן האלגוריתם לא פולינומי ולא ניתן להוכיח האם $COMPOSITES \in P$. נשים לב אבל שחוקרים מצאו אלגוריתם שכן מוכיח ש $COMPOSITES \in P$ אבל לא נלמד על זה.

<u>טענה</u>: $COMPOSITES \in NP$.

<u>רעיון ההוכחה</u>: זמן הריצה צריך להיות פולינומיאלי <u>בכמות הספרות</u> המייצגות את המספר n. נשים לב ש: $\log(n) = |< n >|$ כי זו מחרוזת בינארית ולכן נתעניין כאן במ"ט שרצות בזמן פולינומי ב-$\log(n)$.

<u>הוכחה - דרך 1</u>: נגדיר יחס $R_C = \{(n,d) \in \mathbb{N} \times \mathbb{N} |\ d\ divides\ n\ and\ 1 < d < n\}$.

1. <u>היחס חסום פולינומית</u>: כי $d < n$ ולכן קידוד d לא גדול מקידוד n. כלומר $\log(d) \leq \log(n)$.
2. <u>היחס ניתן להכרעה פולינומית</u>: קיים מאמת שמחלק מספרים טבעיים ביעילות ובודק אם יש שארית.
3. $n \in COMPOSITES$ קיים $1 < d < n$ המחלק את n ללא שארית $\Leftrightarrow \exists d:\ (n,d) \in R_C$.

<u>הוכחה - דרך 2</u>: נגדיר מ"ט א"ד M על קלט $< n >$:

1. תנחש מספר טבעי d כך ש: $1 < d < n$.
2. תבדוק האם d מחלק את n.
3. אם כן, <u>תקבל</u>. אחרת, <u>תדחה</u>.

<u>סיבוכיות זמן ריצה</u>: המכונה פולינומית, כי ניתן לנחש d ולבדוק חלוקה בזמן פולינומי.

<u>הוכחת נכונות</u>: נוכיח כי $COMPOSITE = L(M)$.

- $n \in COMPOSITES \leftarrow$ קיים d שמחלק את $n \leftarrow$ קיים ניחוש ל-d שעבורו קיים מסלול חישוב ש-M מגיע למצב מקבל $\leftarrow n \in L(M)$.
- $n \notin COMPOSITES \leftarrow$ לכל d בתחום, לא קיים מסלול חישוב עבור M שמגיע למצב מקבל $\leftarrow n \notin L(M)$.

השפה PRIMES

נתונה השפה $PRIMES = \{< n > \mid n\ is\ prime\}$.

שאלה: האם $PRIMES \in P$?

תשובה: מכיוון ש: $COMPOSITES \in NP$ אז גם $PRIMES \in P$ (כי P סגורה למשלים).

הוכיחו גם בעבר כי $PRIMES \in NP$.

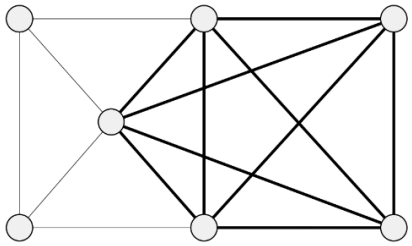
הערה: שפה שגם היא וגם המשלימה שלה שייכות ל-NP אז היא "חשודה" כשייכת גם ל-P.

קליקה בגרף - CLIQUE

הגדרה: קליקה בגרף לא מכוון G היא קבוצה של צמתים שיש קשת בין כל שניים מהם.

נתונה השפה: $CLIQUE = \{< G, k > \mid G\ is\ an\ undirected\ graph\ with\ a\ k - clique\}$.

בציור: גרף שמכיל 5-קליקה.



טענה: $CLIQUE \in NP$.

רעיון ההוכחה: הקליקה היא "תעודת השייכות".

הוכחה: נוכיח כי V הוא המאמת של $CLIQUE$.

V על קלט $<< G, k >, c >$ תבצע:

1. תבדוק האם c הוא תת-גרף עם k קודקודים ב-G.
2. תבדוק האם G מכילה את כל הצלעות שמחברות את הקודקודים בתת הגרף c.
3. אם שלבים 1 ו-2 הצליחו, קבל. אחרת, דחה.

הוכחה אחרת: נציג אלגוריתם N א"ד בזמן פולינומי שמכריע את $CLIQUE$.

המכונה N על הקלט $< G, k >$ תבצע:

1. תנחש תת קבוצה c של k קודקודים בגרף G.
2. תבדוק האם G מכילה את כל הצלעות שמחברות את הקודקודים בתת הגרף c.
3. אם שלב 2 הצליח, קבל. אחרת, דחה.

שאלה: האם $CLIQUE \in P$?

פתרון: לא ידוע.

שאלה: האם IS היא המשלימה של $CLIQUE$?

תשובה: זה לא בדיוק המשלים שלו.... הגרפים כן משלימים אבל השפות הנתונות לא משלימות.

למה? כי גרף קליקה בגודל k אז המשלים שלו הוא גרף עם קבוצה בלתי תלויה בגודל k.

הגרפים $< G, k > \in CLIQUE$ ו-$< G, k > \in IS$ הם משלימים אחד של השני. אבל לא השפות עצמם.

קבוצה בלתי תלויה - IS

הגדרה: בתורת הגרפים, **קבוצה בלתי תלויה** (IS - Independent set) היא קבוצת קודקודים בגרף, אשר אין זוג מביניהם המחוברים ישירות דרך קשת אחת.

נתונה השפה: $IS = \{< G, k > \mid G\ is\ a\ graph\ that\ contains\ an\ independent\ set\ of\ size\ k\}$.

תזכורת: הראנו כבר שמתקיים $IS \in NP$.

שאלה: האם $IS \in P$?

תשובה: לא ידוע.

הערה: נשים לב היטב לקלט ב-2 הבעיות האחרונות (גרף G ומספר טבעי k ולא רק גרף G).

$IS \in R$: כי אפשר לעבור על $\binom{n}{k}$ תתי קבוצות ובכל אחת לוודא שאין צלעות $\binom{k}{2}$.

זמן ריצה: $\Theta(n^k \cdot k^2)$.

הוכחנו שהיא ב-R, אם עברנו על כל האפשרויות ולא מצאנו קבוצה ב"ת אז $\Theta(n^k \cdot k^2)$ וזה נראה פולינום.

אז השאלה היא למה לא ידוע האם $IS \in P$? - הרי כתוב כאן פולינום בחזקה קבועה.

הבעיה: כי $1 \leq k \leq n$, אמנם k הוא מספר קבוע אבל הוא יכול להיות פונקציה של n.

מאינפי: $n^k \cdot k^2$ מקבל מקסימום עבור $k = \frac{n}{2}$ ולכן $\Theta(n^{n/2} \cdot (n/2)^2)$ זה אקספוננציאלי. **???WTF**

במילים אחרות: בגלל ש: $n \to \infty$ ו-k חסום עד n כולל אז יכול להיות מקרה בו $k = n$ ולכן $n^k = n^{n/2} \to \infty$ כלומר זה לא יכול להיות פולינומי.

השפה SUBSET-SUM

נתונה השפה הבאה:

$$SUBSET-SUM=\{<S,t>\mid S=\{x_1,..,x_n\},\ \exists\{y_1,..,y_k\}\subseteq\{x_1,...,x_n\}\ s.t.\ \Sigma y_i=t\}$$

<u>השפה במילים</u>: הקידוד $<S,t>$ בשפה אם לקבוצה S קיים תת קבוצה $\{y_1,..,y_k\}$ כך שסכומה הוא t.

<u>דוגמה</u>: $<\{4,11,16,21,27\},\ 25>\in SUBSET-SUM$ בגלל ש: $4+21=25$.

<u>טענה</u>: $SUBSET-SUM\in NP$.

<u>רעיון ההוכחה</u>: תת הקבוצה היא "תעודת השייכות".

<u>הוכחה</u>: נוכיח כי V הוא המאמת של $SUBSET-SUM$.

המוודא V על הקלט $<<S,t>,\ c>$ מבצע:

1. בדוק האם c הוא אוסף מספרים שסוכמו שווה ל-t.
2. בדוק האם S מכיל את כל המספרים ב-c.
3. אם שלב 1 ו-2 הצליחו, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>הוכחה אחרת</u>: נציג אלגוריתם N א"ד בזמן פולינומי שמכריע את השפה $SUBSET-SUM$.

המכונה N על הקלט $<S,t>$ תבצע:

1. נחש תת קבוצה c של מספרים מתוך S.
2. בדוק האם c הוא אוסף מספרים שסוכמו שווה ל-t.
3. אם שלב 1 הצליח, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>שאלה</u>: האם היא שייכת ל-P $SUBSET-SUM\in P$?

<u>תשובה</u>: לא ידוע.

הבהרה של כללי המשחק

<u>שאלה</u>: משהו פה לא ברור. אם מישהו מספק לנו את ההוכחה לשייכות של המילה לשפה, אז האם לא כל שפה שייכת ל-NP?

- האם לא תמיד אפשר לספק "תעודת שייכות" קצרה לכל מילה בשפה?
- הכוונה ב"קצרה" היא בעלת אורך פולינומיאלי באורך המילה.

<u>תשובה</u>: לא כל שפה אפשר להכניס פנימה. הסתכלו במשלימה של $CLIQUE$ או של $HAMPATH$ או של IS.

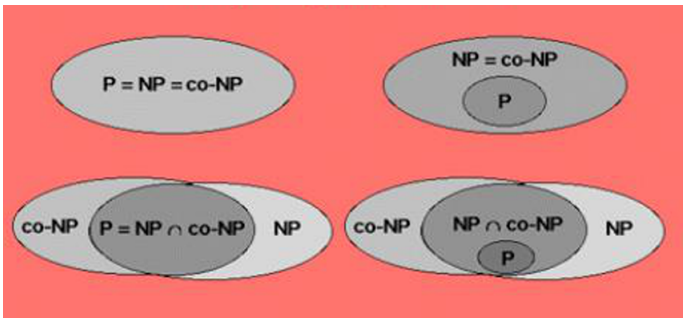
האם תוכלו להראות שהן שייכות ל- NP? המשלים של $CLIQUE$ אומר שכל הקבוצות ב-G שאין להם קליקה בגודל k. כלומר צריך את כל הקבוצות בגודל k ולוודא שכל אחת ואחת מהן היא לא קליקה. לכן ה-"עד" הוא קבוצה של קבוצות ולכן זה כבר אקספוננציאלי - לא חסום פולינומית.

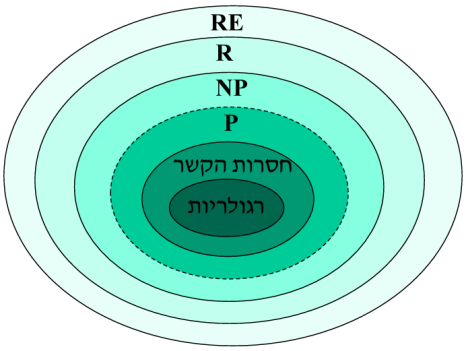
שאלות פתוחות

- לא ידוע האם $P = NP$ או $P \subset NP$. האם יש שוויון בין המחלקות או שיש הכלה ממש?
  רוב מוחלט של החוקרים סבורים ש: $P \neq NP$.
- לא ידוע האם $NP = coNP$
  הסבירו היטב את משמעות השוויון הזה! (**להשלים**).
- לא ידוע האם $P = NP \cap coNP$.
  הסבירו היטב מהי המחלקה $NP \cap coNP$. הראו ש: $P \subseteq NP \cap coNP$. (**להשלים**).

תמונת עולם של המחלקות

האם כל ההכלות בין המחלקות הן הכלות ממש? (**להשלים**).





תכונות סגור של NP

<u>תרגיל</u>: הוכיחו שהמחלקה NP סגורה ל:

1. איחוד
2. חיתוך
3. שרשור
4. איטרציה

- לא ידוע האם NP סגורה למשלים.
- לא ידוע האם $NP = coNP$.

שאלה 1 - תרגול 5 - מעגל אוילר

נתונה השפה: $EC = \{< G > \mid G\ has\ an\ Euler\ cycle\}$.

<u>הגדרה</u>: **מסלול אוילר** הוא מסלול בגרף העובר בכל קשת בו בדיוק פעם אחת. **מעגל אוילר** הוא מסלול אוילר מעגלי (מתחיל ונגמר באותו צומת).

<u>תזכורת (מקורס אלגוריתמים)</u>: בגרף קשיר קיים מעגל אוילר אם ורק אם כל הדרגות זוגיות.

<u>טענה</u>: $EC \in P$.

<u>רעיון ההוכחה</u>: אם נרצה לחפש את המעגל עצמו, אז אנחנו נצטרך לבצע חיפוש שלם בכל הגרף כדי למצוא את המעגל. אבל אם נעבור על כל הפרמוטציות אז האלגוריתם יהיה בזמן ריצה אקספוננציאלי ואנחנו רוצים בזמן פולינומי. אנחנו נרצה לבנות את האלגוריתם שלנו כאשר גודל הקלט שלנו יהיה מטריצת שכנויות של G. למה אנחנו יכולים להשתמש במטריצת שכנויות? - אם גודל הקלט של השפה $< G >$ נקבע על סמך מספר הצמתים n, אז אפשר להשתמש בקלט אחר מסוג טבלת שכנויות $nxn = n^2$ שגם בגודל פולינומי.
בהרצאה למדנו שאם קיים יחס פולינומי בין קלט a לקלט b אז אפשר להשתמש בקלט b במקום a.
לכל קודקוד בטבלה (לכל שורה) נסכום את הדרגות (השכנים) וכל פעם נבדוק האם הסכום זוגי.

<u>הוכחה</u>: נראה מכונת טיורינג M_{EC} דטרמיניסטית עבור EC, שרצה בזמן חסום פולינומית ביחס לקלט שלה.

מכונת טיורינג M_{EC} על קלט $< G >$ תבצע:

1. עבור על כל הקודקודים $v_i \in V(G)$:
   a. עבור כל קודקוד v_i, בדוק מה מספר השכנים שלו וסמן זאת ב-c.
   b. בדוק האם c הוא מספר זוגי. אם לא, <u>דחה</u>.
   c. אחרת, עבור לקודקוד v_{i+1}.
2. <u>קבל</u>. (נוכל לקבל פה כי עברנו על כל הצמתים ולא דחינו, כלומר כל הדרגות זוגיות).

<u>סיבוכיות זמן ריצה</u>: נאמר שהקלט הוא $n = |V(G)|$.

- שלב 1 - מעבר על כל הצמתים עולה לנו $O(n)$.
- שלב a - לכל צומת v_i נעבור על כל השכנים - עולה לנו $O(n)$.

סך הכל זמן הריצה היא $O(n^2)$ שאכן בזמן פולינומי (עבור $p(n) = n^2$).

<u>הוכחת נכונות</u>: צריך להוכיח כי $EC = L(M_{EC})$.

- $< G > \in EC$ ← קיים מעגל אוילר בגרף G ← כל הצמתים בגרף בעלי דרגות זוגיות ← M_{EC} תעבור בלולאה על כל הצמתים ← לאחר שתסיים לעבור על כולם, תתקבל ← $M_{EC}(< G >) == 1$ ← $< G > \in L(M_{EC})$.
- $< G > \notin EC$ ← לא קיים מעגל אוילר בגרף G ← קיימת לפחות צומת אחת בגרף בעלת דרגה אי-זוגית ← המכונה M_{EC} תעבור על הצמתים עד שתפגוש בצומת עם דרגה אי זוגית ← $M_{EC}(< G >) == 0$ ← $< G > \notin L(M_{EC})$.

שאלה 2 - תרגול 5 - מעגל המילטון

נתונה השפה: $HC = \{< G > \mid G\ has\ a\ Hamiltonian\ cycle\}$.

<u>הגדרה</u>: **מסלול המילטוני** הוא מסלול בגרף מכוון או בלתי מכוון העובר בכל צומת בדיוק פעם אחת.
מעגל המילטוני הוא מסלול בגרף העובר בכל צומת פעם אחת פרט לצומת שממנו יצא (ואז הוא עובר בו בדיוק פעמיים - בהתחלה ובסוף).

<u>שאלה</u>: האם ניתן להוכיח ש: $HC \in P$.
<u>תשובה</u>: נכון להיום, לא ידוע האם $HC \in P$. זה לא אומר שזה לא שייך, ייתכן שלא מצאו אלגוריתם לבעיה.

<u>רעיון</u>: אם נרצה לעבור על כל הפרמוטציות (חיפוש שלם) אז זה על-פולינומי (אקספוננציאלי).
במעגל אוילר ניצלנו את השיטה של הדרגות הזוגיות, אבל נכון להיום אין לנו שיטה דומה עבור מעגל המילטון.
נשים לב ש: $HC \in R$ משום שניתן לעבור על כל המעגלים האפשריים בגרף ולבדוק האם הם מתאימים.
פורמלית, נעבור על כל הסידורים של צמתי הגרף $v_1,...,v_n$ ולכל סידור כזה נבדוק אם לכל i מתקיים $(v_i, v_{i+1}) \in E(G)$ וכן כי $(v_n, v_1) \in E(G)$. אם מצאנו סידור שעונה על התנאי אז מצאנו מעגל המילטון.
נניח שגודל הקלט הוא כמספר הצמתים n, לכן האלגוריתם שתיארנו יקח $n \cdot (n-1) \cdot ... \cdot 1 = n!$ צעדים כלומר $p(n) = n!$ וזה זמן שאינו פולינומי בגודל הקלט ולכן למרות ש: $HC \in R$, לא הצלחנו להוכיח כי $HC \in P$. אך זה לא אומר ש: $HC \notin P$, פשוט אף אחד לא מצא אלגוריתם שפותר את הבעיה ולכן לא ידוע.

שאלה 3 - תרגול 5 - מעגל המילטון

<u>טענה</u>: $HC \in NP$.

<u>רעיון ההוכחה</u>: מ"ט א"ד פולינומית מסוגלת להכריע שפות שאיננו יודעים אם הן ב-P. ראינו בשאלה קודמת כי אנחנו לא יודעים האם $HC \in P$. כלומר ראינו שנכון להיום, לא ידוע אם קיימת מכונה דטרמיניסטית שפותרת את הבעיה HC בזמן פולינומי. מצד שני, אם יהיה לנו אפשרות לנחש את הפרמוטציה שתייצג את הסידור הנכון בגרף עבור מעגל המילטון אז ניתן יהיה לפתור בזמן פולינומי. נציג מכונה א"ד פולינומית עבור HC.

<u>הוכחה</u>: המכונה M_{HC} על הקלט $< G >$ תבצע:

1. מנחשת פרמוטציה (מסלול) על הצמתים ובודקת שאכן מהווה מעגל המילטון.
2. אם כן, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>סיבוכיות זמן ריצה</u>: אורך המעגל שצריך לנחש הוא ליניארי (מסלול בעץ החישוב) כלומר $O(n)$. צריך לבדוק האם הצמתים שניחשנו שונים זה מזה, ושבין כל שני צמתים עוקבים בסידור קיימת קשת - עולה לנו $O(n)$.
לכן, שתי הבדיקות הללו ניתנות לביצוע בזמן פולינומי.

<u>הוכחת נכונות</u>: צריך להוכיח ש: $HC = L(M_{HC})$.

- $< G > \in HC$ ← קיים ב-G מעגל המילטון ← ניתן לתאר את המעגל ההמילטוני כסידור של צמתים ← M_{HC} תנחש סידור זה והבדיקה שלה תצליח ← קיים מסלול חישוב מקבל עבור G ← $< G > \in L(M_{HC})$.

- $HC \notin < G >$ ← לא קיים ב-G מעגל המילטון ← לא קיים אף סידור של צמתים המתאר מעגל המילטוני ← M_{HC} לא תצליח בניחוש ← לא קיים אף מסלול חישוב מקבל עבור G ← $< G > \notin L(M_{HC})$.

ומכאן $HC \in NP$ כנדרש.

∎

שאלה 4 - תרגול 5

נתונה השפה $L_1 = \{< G > \mid \exists C \subseteq V(G)\ s.t.\ |C| = 11\ and\ C\ is\ a\ clique\}$.

<u>טענה</u>: $L_1 \in NP$.

<u>רעיון ההוכחה</u>: נבנה מכונה א"ד N_1 בזמן ריצה פולינומית שמכריעה את השפה L_1.

<u>הוכחה</u>: <u>בניית המכונה</u>: המכונה N_1 על קלט $< G >$ תבצע:

1. ננחש קבוצה של קודקודים בגודל 11.
2. לכל זוג קודקודים בקבוצה, נבדוק שיש ביניהם צלע, ואם לא - <u>נדחה</u>.
3. אם כל הבדיקות עברו, <u>נקבל</u>.

<u>הוכחת נכונות הבניה</u>:

- $< G > \in L_1$ ← קיימת קליקה בגודל 11 בגרף G ← קיים ניחוש בו כל הצלעות קיימות ← N_1 בוחר בניחוש הנכון ומקבל ← $< G > \in L(N_1)$.
- $< G > \notin L_1$ ← לא קיים קליקה בגודל 11 בגרף G ← לכל ניחוש, לפחות צלע אחת תהיה חסרה ← לכל ניחוש, נדחה ← $< G > \notin L(N_1)$.

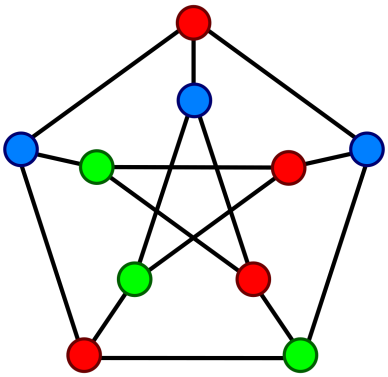
<u>סיבוכיות זמן הריצה</u>:

- שלב 1 - הניחוש נקבע לפי גודל הניחוש ולכן זה עולה לנו $O(1) \approx O(11)$.
- שלב 2 - צריך לבדוק על כל זוג קודקודים ולכן זה עולה לנו $O(1) \approx O(\binom{11}{2})$.

סך הכל $O(1)$, פולינומי.

∎

שאלה 5 - תרגול 5 - צביעה של גרף



<u>הגדרה</u>: בתורת הגרפים, גרף n-צביע הוא גרף שאפשר לצבוע את הקודקודים שלו ב-n צבעים, כך ששני קודקודים סמוכים אינם צבועים באותו צבע. נתונה השפה:

$$L_2 = \{(< G >, k) \mid V(G)\ can\ be\ colored\ with\ k\ colors\ s.t.\ \forall_{(u,v)\in E(G)} color(v) \neq color(u)\}$$

<u>במילים</u>: הצמד (קידוד של גרף, מספר k טבעי) שייך לשפה L_2 אם קיים צביעה של k צבעים לגרף G, כך שלכל צלע, שני צמתיו בצבע שונה.

<u>טענה</u>: $L_2 \in NP$.

<u>הוכחה</u>: <u>בניית המכונה</u>: נבנה N_2 מ"ט א"ד פולינומית המכריעה את השפה L_2.

המכונה N_2 על קלט $(< G >, k)$ תבצע:

1. לכל צומת:
   a. נחש "צבע" בין 1 ל-k.
2. לכל צלע:
   a. בדוק ששני הצמתים שלה צבועים בצבע שונה, אם לא, <u>דחה</u>.
3. אם כל הבדיקות עברו, <u>קבל</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $L(N_2) = L_2$.

- $(< G >, k) \in L_2$ ← קיים צביעה של k צבעים לגרף G, כך שלכל צלע, שני צמתיו בצבע שונה ← קיים ניחוש שעבורו N_2 בוחר נכון ומקבל ← $(< G >, k) \in L(N_2)$.
- $(< G >, k) \notin L_2$ ←לא קיים בגרף G צביעה כזאת ← לכל ניחוש, קיים לפחות צלע אחת שצמתיו בצבע זהה ← $(< G >, k) \notin L(N_2)$.

<u>סיבוכיות זמן הריצה</u>:

- שלב 1:
  - מעבר על כל $n = |V(G)|$ צמתים עולה לנו $O(n)$.
  - ניחוש הצבע עולה לנו $O(\log k)$ כי אם יש לנו בסה"כ k צבעים אז נצטרך $\log k$ ביטים על מנת לקודד צבע.
  - סה"כ לשלב 1 עולה לנו $O(n \log k)$.
- שלב 2:
  - לעבור על כל הצלעות עולה לנו $O(n^2)$ - כי זה המקרה הגרוע של מעבר על מטריצת שכנויות.
  - בדיקה לכל צלע עולה לנו $O(1)$.

סה"כ עולה לנו $O(n^2 + n \log k)$, פולינומי ולכן $L_2 \in NP$ כנדרש.

∎

שאלה 6 - תרגול 5

נתונה השפה: $L_3 = \{(< G >, f, d) \mid \exists I \subseteq V(G),\ |I| = f,\ \forall v \in V(G) :\ dist(v, I) \leq d\}$.

במילים: קיימת קבוצה של צמתים I, כך שכל הצמתים בגרף נמצאים במרחק לכל היותר d מקודקוד בקבוצה I.

טענה: $L_3 \in NP$.

הוכחה: בניית המכונה: נבנה מ"ט א"ד N_3 בזמן פולינומי שמכריעה את השפה L_3.

המכונה N_3 על קלט $(< G >, f, d)$ תבצע:

1. נחש קבוצה של f קודקודים.
2. לכל צומת v בגרף:
   a. תריץ החל ממנו BFS עד שתגיע לקודקוד בקבוצה. אם המרחק הוא יותר מ-d - דחה.
3. אם כל הבדיקות עברו - קבל.

הוכחת נכונות הבניה: נוכיח כי $L(N_3) = L_3$.

- $(< G >, f, d) \in L_3$ ← קיימת קבוצה I של f צמתים, כך שכל הצמתים בגרף נמצאים במרחק לכל היותר d מקודקוד בקבוצה I ← קיים ניחוש של קבוצה כזו ← N_3 בוחר בניחוש הנכון ומקבל ← $(< G >, f, d) \in L(N_3)$.
- $(< G >, f, d) \notin L_3$ ← לא קיימת קבוצה I של f צמתים, כך שכל הצמתים בגרף נמצאים במרחק לכל היותר d מקודקוד בקבוצה I ←לכל ניחוש, קיים צומת בגרף שרחוק יותר ב-d מכל הצמתים בקבוצה I ← לכל ניחוש, N_3 דוחה ← $(< G >, f, d) \notin L(N_3)$.

סיבוכיות זמן ריצה:

1. שלב 1 - עולה כגודל הניחוש: $O(f) = O(n)$ כאשר $|V(G)| = n$.
2. שלב 2 -
   a. לעבור על כל צומת בגרף עולה לנו $O(n)$.
   b. לבצע BFS עולה לנו $O(n^2)$ כי זה הסיבוכיות של BFS לגרף המיוצג במטריצת שכנויות.
   c. סה"כ עבור שלב 2 נקבל $O(n \cdot n^2) = O(n^3)$.
3. שלב 3 - קבלה עולה לנו $O(1)$.

סה"כ קיבלנו $O(n^3)$ שזה פולינומי ולכן $L_3 \in NP$ כנדרש.

∎

שאלה 7 - תרגול 5 - מסלול קצר ביותר בגרף

נתונה השפה:

$$L_4 = \{(< G >, k) |\ The\ shortest\ path\ in\ G\ (between\ any\ two\ nodes)\ is\ of\ length\ < k\}$$

במילים: המסלול הקצר ביותר בגרף G בין זוג קודקודים כלשהם הוא לכל היותר באורך $k - 1$.

טענה: $L_4 \in NP$.

הוכחה: בניית המכונה: נבנה N_4 מ"ט א"ד פולינומית המכריעה את השפה L_4.

המכונה N_4 על הקלט $(< G >, k)$ תבצע:

1. נחש מסלול.
2. בדוק שאורכו קטן מ-k.
3. אם כן, קבל. אחרת, דחה.

הוכחת נכונות הבניה: נוכיח כי $L_4 = L(N_4)$.

- $(< G >, k) \in L_4$ ← המסלול הקצר ביותר בגרף G בין זוג קודקודים כלשהם הוא לכל היותר באורך $k - 1$ ← קיים ניחוש למסלול כזה ← N_4 בוחר בניחוש הנכון ומקבל ← $(< G >, k) \in L(N_4)$.
- $(< G >, k) \notin L_4$ ← המסלול הקצר ביותר בגרף G הוא לפחות באורך k ← לכל ניחוש, נבחר במסלול שלפחות באורך k ← לכל ניחוש, N_4 דוחה ← $(< G >, k) \notin L(N_4)$.

סיבוכיות זמן ריצה:

1. שלב 1 - אורך המסלול חסום בכמות הצמתים ולכן ניחוש המסלול עולה לנו $O(n)$.
2. שלב 2 - חישוב אורך המסלול: מעבר לאורך כל הצלעות ולכן עולה לנו $O(n)$.
3. שלב 3 - בדיקת קבלה /דחיה עולה לנו $O(1)$.

סה"כ קיבלנו $O(n)$ שזה פולינומי ולכן $L_4 \in NP$ כנדרש.

∎

שאלה 8 - תרגול 5

נתונות השפות שראינו:

$$L_1 = \{< G > \mid \exists C \subseteq V(G)\ s.t.\ |C| = 11\ and\ C\ is\ a\ clique\}$$

$$L_2 = \{(< G >, k) \mid V(G)\ can\ be\ colored\ with\ k\ colors\ s.t.\ \forall_{(u,v)\in E(G)} color(v) \neq color(u)\}$$

$$L_3 = \{(< G >, f, d) \mid \exists I \subseteq V(G),\ |I| = f,\ \forall v \in V(G):\ dist(v, I) \leq d\}$$

$$L_4 = \{(< G >, k) |\ The\ shortest\ path\ in\ G\ (between\ any\ two\ nodes)\ is\ of\ length\ < k\}$$

שאלה: אילו שפות אנחנו יכולים להראות שהן ב-P?

תשובה: השפות L_1 ו- L_4 .

שאלה: למה $L_1 = \{< G > \mid \exists C \subseteq V(G)\ s.t.\ |C| = 11\ and\ C\ is\ a\ clique\} \in P$?

תשובה: אפשר לבנות מכונה דטרמיניסטית שתכריע את השפה L_1 בזמן פולינומי. מספר כל הקבוצות האפשריות בגודל 11 מתוך n קודקודים הוא $\binom{n}{11} \leq n^{11}$ ולכן אנחנו יכולים בזמן פולינומי לבדוק את כל תתי הקבוצות בגודל 11 (כי זה דטרמיניסטי) כי רק $O(n^{11})$ כאלו.

שאלה: למה $L_4 = \{(< G >, k) \mid\ The\ shortest\ path\ in\ G\ is\ of\ length\ < k\} \in P$?

תשובה: כמה מסלולים אפשריים יש לנו בגרף G?: יש לנו n^2 מסלולים בין כל זוג קודקודים, ואם נעבור על כל המסלולים האלה, אפשר למצוא את המסלול המינימלי ביותר בגרף G ובסופו לוודא שהוא קטן מספיק מ-k. לכן קיימת מכונה דטרמיניסטית בזמן פולינומי שתכריע את השפה L_4.

שאלה 9 - תרגול 5

עבור מ"ט M, נגדיר את השפה: $L_{4,M} = \{x \mid M\ has\ at\ least\ 4\ accepting\ paths\ running\ on\ x\}$.

נגדיר מחלקת שפות: $NP_4 = \{L \mid \exists M\ nondeterministic\ polynomial\ TM\ s.t.\ L_{4,M} = L\}$.

הוכיחו: $NP \subseteq NP_4$.

רעיון ההוכחה: נבנה מכונה M_L' של שפה ב-NP_4 שתפעיל מכונה M_L של שפה ב-NP, כך שבכל פעם שיהיה לנו מסלול מקבל ב-M_L, המכונה M_L' תפצל את המסלול הזה ל-4 תתי מסלולים כך שכל אחד מהם הוא גם מסלול מקבל.

הוכחה: תהי $L \in NP$. צריך להוכיח כי $L \in NP_4$.

$L \in NP$ ולכן קיימת M_L מ"ט א"ד פולינומית המכריעה את השפה L.

<u>בניית המכונה</u>: נבנה את M_L' מ"ט א"ד חדשה:

המכונה M_L' על קלט x תבצע:

1. מריצה את M_L על x.
2. אם דחתה - <u>דוחה</u>.
3. אם M_L קיבלה את x:
4. מנחשת ביט.
5. מנחשת ביט.
6. <u>מקבלת</u>.

<u>סיבוכיות זמן הריצה</u>: קל לראות ש: M_L' פולינומית. הרצה של M_L על x היא פולינומית כי $L \in NP$, ושאר הניחושים של M_L' הם בזמן קבוע.

הוכחת נכונות הבניה: עבור שפה $L \in NP$ ושפה $L_{4,M_L'} \in NP_4$, נוכיח כי $L = L_{4,M_L'}$.

- אם $x \notin L \leftarrow M_L$ תעצור ותדחה בכל מסלול חישוב (לפי הגדרה של מ"ט א"ד מכריעה) $\leftarrow M_L'$ תעצור ותדחה בכל מסלול חישוב $\leftarrow x \notin L_{4,M_L'}$.
- אם $x \in L \leftarrow$ קיים מסלול חישוב מקבל שבו M_L תעצור ותקבל $\leftarrow$ ל- M_L' יש ארבעה מסלולי חישוב מקבלים, עבור ניחושי הביטים 00,01,10,11 $\leftarrow x \in L_{4,M_L'}$

הוכחנו כי $L = L_{4,M_L'}$ ומאחר ו- $L_{4,M_L'} \in NP_4$ אז גם $L \in NP_4$.

ומכאן $NP \subseteq NP_4$ כנדרש.

∎

שאלה 10 - תרגול 5 - P סגורה למשלים

טענה: אם $L \in P$ אז גם $\bar{L} \in P$.

הוכחה: מכיוון ש $L \in P$, אזי יש לה מ"ט M דטרמיניסטית פולינומית שמכריעה אותה.

בניית המכונה: נבנה מ"ט M' דטרמיניסטית פולינומית מכריעה עבור $\bar{L}$.

המכונה M' על קלט x תבצע:

1. מריצה את M על x.
2. אם M קיבלה, תדחה.
3. אם M דחתה, תקבל.

הוכחת נכונות הבניה: נוכיח כי $\bar{L} = L(M')$.

- $x \in \bar{L} \leftarrow M$ דוחה $\leftarrow M'$ תקבל $\leftarrow x \in L(M')$.
- $x \notin \bar{L} \leftarrow M$ מקבלת $\leftarrow M'$ דוחה $\leftarrow x \notin L(M')$.

סיבוכיות זמן ריצה: מאחר ו-M מ"ט מכריעה ופולינומית אז גם M' פולינומית.

∎

שאלה 11 - תרגול 5

טענה: $P \subseteq coNP$.

הוכחה: אם $L \in P$ אז לפי סגירות למשלים גם $\bar{L} \in P$. מאחר ו-$P \subseteq NP$ אזי $\bar{L} \in NP$ ומכאן $\bar{\bar{L}} = L \in coNP$.

∎

שאלה 12 - תרגול 5 - P סגורה לחיתוך

<u>טענה</u>: אם $L_1, L_2 \in P$, אז גם $L_1 \cap L_2 \in P$.

<u>הוכחה</u>: מכיוון ש: $L_1, L_2 \in P$ אז קיימות להן מ"ט דטרמיניסטיות פולינומיות מכריעות M_1, M_2.

<u>בניית המכונה</u>: נבנה M' מ"ט דטרמיניסטית פולינומית מכריעה עבור השפה $L_1 \cap L_2$.

המכונה M' על הקלט x תבצע:

1. תסמלץ את M_1 על x.
2. תסמלץ את M_2 על x.
3. אם שתיהן מקבלות, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $L(M') = L_1 \cap L_2$.

- $x \in L_1 \cap L_2$ ← M_1 מקבל על x וגם M_2 מקבל על x ← M' מקבל ← $x \in L(M')$.
- $x \notin L_1 \cap L_2$ ← M_1 דוחה על x או M_2 דוחה על x ← M' דוחה ← $x \notin L(M')$.

<u>סיבוכיות זמן ריצה</u>: מאחר ו-M_1, M_2 הן מכונות מכריעות ופולינומיות אזי גם M' פולינומית.

∎

שאלה 13 - תרגול 5 - NP סגורה לשרשור

<u>טענה</u>: אם $L_1, L_2 \in NP$ אזי גם $L_1 \cdot L_2 \in NP$.

<u>הוכחה</u>: מכיוון ש: $L_1, L_2 \in NP$ אזי קיימות להן מ"ט א"ד פולינומיות מכריעות M_1, M_2.

נבנה את M' מ"ט א"ד פולינומית שמכריעה את השפה $L_1 \cdot L_2$.

המכונה M' על קלט x תבצע:

1. תנחש פירוק של x מהצורה $x = wu$.
2. תסמלץ את M_1 על w.
3. תסמלץ את M_2 על u.
4. אם שתיהן קיבלו, <u>קבל</u>. אחרת, אם לפחות אחת מהן דחתה, <u>דחה</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $L(M') = L_1 \cdot L_2$.

- $x \in L_1 \cdot L_2$ ← קיים פירוק $x = wu$ כך ש: $w \in L_1 \wedge u \in L_2$ ← קיים ניחוש כזה $M_1(w) = 1 \wedge M_2(u) = 1$ ← M' תקבל ← $x \in L(M')$.
- $x \notin L_1 \cdot L_2$ ← לכל פירוק $x = wu$ כך ש: $w \notin L_1 \vee u \notin L_2$ ← לכל ניחוש, $M_1(w) = 0 \vee M_2(u) = 0$ ← M' תדחה ← $x \notin L(M')$.

<u>סיבוכיות זמן ריצה</u>: המכונות M_1, M_2 פולינומיות ולכן גם M' פולינומית.

∎

שאלה 14 - תרגול 5 - P סגורה לשרשור

<u>טענה</u>: אם $L_1, L_2 \in P$ אזי גם $L_1 \cdot L_2 \in P$.

<u>הוכחה</u>: מכיוון ש: $L_1, L_2 \in P$ אזי קיימות להן מ"ט דטרמיניסטיות פולינומיות מכריעות M_1, M_2.

נבנה את M' מ"ט דטרמיניסטית פולינומית שמכריעה את השפה $L_1 \cdot L_2$.

המכונה M' על קלט x תבצע:

1. לכל פירוק של x מהצורה $x = wu$:
   a. תסמלץ את M_1 על w.
   b. תסמלץ את M_2 על u.
   c. אם שתיהן קיבלו, <u>קבל</u>.
2. אם כל הפירוקים לא הצליחו - <u>דחה</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $L(M') = L_1 \cdot L_2$.

- $x \in L_1 \cdot L_2 \leftarrow$ קיים פירוק $x = wu$ כך ש: $w \in L_1 \wedge u \in L_2 \leftarrow$ קיים פירוק כזה כך ש:
  $M_1(w) = 1 \wedge M_2(u) = 1 \leftarrow M'$ תקבל $\leftarrow x \in L(M')$.
- $x \notin L_1 \cdot L_2 \leftarrow$ לכל פירוק $x = wu$ כך ש: $w \notin L_1 \vee u \notin L_2 \leftarrow$ לכל פירוק,
  $M_1(w) = 0 \vee M_2(u) = 0 \leftarrow M'$ תדחה $\leftarrow x \notin L(M')$.

<u>סיבוכיות זמן ריצה</u>: המכונות M_1, M_2 פולינומיות ולכן גם M' פולינומית.

כמה אפשרויות פירוק יש למילה $|x| = n$? - יש לנו $n + 1$ אפשרויות כאלה ולכן $O(n)$ - פולינומי.

לכל אחת מה-n אפשרויות שלנו, נבדוק עבורה את המכונות M_1, M_2 שהם בזמן פולינומי.

∎

שאלה 15 - תרגול 5 - P סגורה לאיטרציה

<u>טענה</u>: אם $L \in P$ אז גם $L^* \in P$.

<u>תזכורת</u>: כאשר הוכחנו ש: R סגורה לאיטרציה, מה שעשינו זה פשוט לעבור על כל החלוקות האפשריות של המילה לתתי מילים.

<u>בעיה</u>: יש כמות אקספוננציאלית של חלוקות כאלה!

<u>שיטה</u>: מתי $x \in L^*$?

1. אם $x = \varepsilon$.
2. אם $x \in L$.
3. אם קיימים מילים $w, u \in L^*$ כך ש: $x = wu$.

נשתמש בתכנון דינאמי על מנת לפתור את הבעיה ☹.

<u>תכנון דינמי</u>:

עבור מילה w, נגדיר את $w_{i,j}$ להיות תת המילה מאינדקס i ועד אינדקס j.

נבנה טבלה Tab המאותחלת ב-F, בה כל תא i, j יחזיק ערך בוליאני שיסמן האם תת-המילה $w_{i,j}$ שייכת ל-L^*.

לכל תא $[i][j]$ בטבלה, נצטרך לבדוק:

1. האם $w_{i,j} \in L$.
2. לבדוק את כל החלוקות לשתי מילים בין הטווח $[i, j)$, ואם באחת מהן שתי המילים ב-L^*.

<u>מה תהיה התשובה הסופית שנקבל בעזרת הטבלה</u>?: התשובה נמצאת בתא $Tab[1][n]$.

למה? - כי זה תת המילה $w_{1,n}$ כלומר זו המילה w כולה שבדיוק עליה נרצה לשאול האם $w \in L^*$.

<u>המכונה המלאה</u>:

	M על קלט w תבצע:
$O(1)$	אם $w = \varepsilon$ - אז <u>נקבל</u>.
$O(n^2)$	נאתחל טבלה Tab בגודל nxn עם הערכים $False$.
$O(n)$	**עבור** $1 \leq i \leq n$:
$O(n)$	**עבור** $i \leq j \leq n$: (אנחנו עוברים החל מ-i, כי אנחנו צריכים חלוקה ל-2 כך שהאינדקס הימני גדול מהשמאלי).
$p(n)$	**נריץ** את M_L על $w_{i,j}$.
$O(1)$	**אם** קיבלה: $Tab[i][j] = True$. (התנאי הראשון עבד, המילה אכן בשפה ונסמן אותה).
	אחרת: (נבדוק את התנאי השני, כלומר נבדוק את כל החלוקות לשתי מילים).
$O(n)$	**עבור** $i \leq k \leq j - 1$: (כלומר נבדוק שאם $w \in L^*$ וגם $u \in L^*$ אז גם $x = wu \in L^*$).
$O(1)$	**אם** $Tab[k + 1][j] = True$ **וגם** $Tab[i][k] = True$: **אז** $Tab[i][j] = True$.
$O(1)$	**אם** $Tab[1][n] = True$, אז <u>נקבל</u>.
	אחרת, <u>נדחה</u>.

<u>זמן הריצה של המכונה שבנינו</u>:$O(n^2 \cdot p(n)) + O(n^2 \cdot n) = O(n^2 \cdot p(n) + n^3)$.

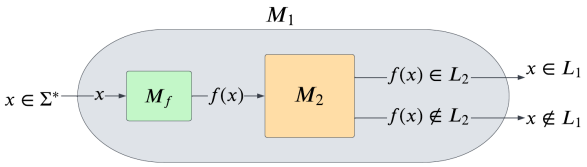
שכמובן שזה בזמן פולינומי ולכן אם $L \in P$ אז גם $L^* \in P$.

רדוקציה פולינומית

תזכורות מתורת החישוביות

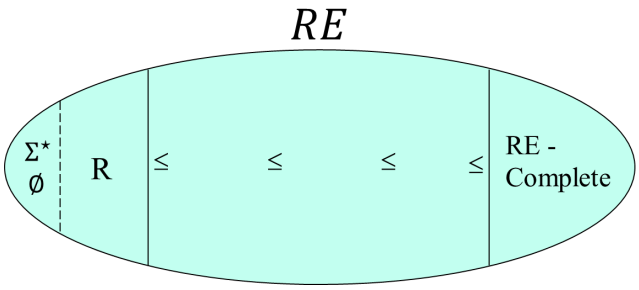
1. **השפות הקלות** במחלקה RE הן השפות הכריעות (R).
   יש רדוקציה מכל שפה $L \in R$ לכל שפה לא טריוויאלית ב-RE.
2. **השפות הקשות** במחלקה RE הן השפות השלמות ב-RE.
   יש רדוקציה מכל שפה $L \in RE$ לכל שפה שלמה ב-RE.

רעיון הרדוקציה:



בפרק זה, באופן כמעט דומה לרדוקציות שראינו עד היום, NP יהיו **הקשים** ו-P יהיו **הקלים** (כי $P \subseteq NP$).

תזכורת: איור של המחלקה RE (חוץ מהשפות הטריוויאליות) משמאל:



הרדוקציות הכלליות לא מתאימות

אם נרצה לבנות משהו דומה במחלקה NP, לא נוכל להשתמש ברדוקציות לא מוגבלות (נרצה משהו פולינומי).כל השפות ב-NP הן כריעות, וממילא ניתנות לרדוקציה לכל שפה ב-NP (פרט לטריוויאליות).
לכן נגדיר סוג מיוחד של רדוקציות - רדוקציות שאפשר לחשב אותן בזמן פולינומיאלי.
אלה רדוקציות שיש מכונה שמחשבת את הרדוקציה בזמן פולינומיאלי בגודל הקלט.
שאלה: האם כל שפה ב-NP ניתנת לרדוקציה פולינומית לכל שפה ב-NP (פרט לטריוויאליות)?
תשובה: **לא ענה על זה בהרצאה.**

רדוקציות בזמן פולינומיאלי

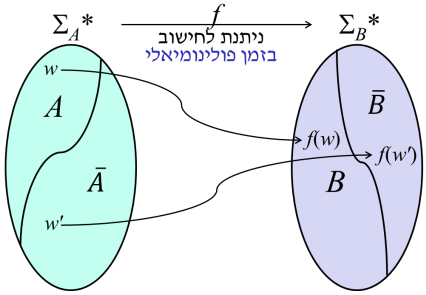
הגדרה: תהיינה A, B שפות מעל האלפבתים Σ_A ו-Σ_B. רדוקציה בזמן פולינומיאלי של A ל-B היא פונקציה מלאה (מוגדרת לכל קלט) הניתנת לחישוב בזמן פולינומיאלי (בגודל הקלט) $f: \Sigma_A^\star \to \Sigma_B^\star$ שמקיימת:

1. אם $w \in A$ אז $f(w) \in B$.
2. אם $w \notin A$ אז $f(w) \notin B$.

בדרך כלל ברדוקציות כאלה נוכיח ש: $w \in A \to f(w) \in B$ וגם $f(w) \in B \to w \in A$.

סימון: אם יש רדוקציה בזמן פולינומיאלי של A ל-B מסמנים $A \leq_P B$.

<u>ציור רדוקציוניסטי</u>: כל שפה ב-R אפשר לעשות רדוקציה לשפה ב-RE. כל שפה ב-P אפשר לעשות רדוקציה לשפה ב-NP. חוץ מ-2 הטריוויאליות $\emptyset, \Sigma^*$. למה? - אם מדובר למשל בשפה $A = \Sigma^*$ אז בפונקצית הרדוקציה אנחנו צריכים לקחת את מה שיש ב-A ל-B וגם לקחת את כל מה שב-$\bar{A}$ ל-$\bar{B}$, אבל ב-$\bar{A}$ אין כלום! (זה נכון גם להפך אם $A = \emptyset$). בין $\emptyset$ ל-Σ^* אפשר לעשות רדוקציה (או להפך), אבל מהשפות $\emptyset, \Sigma^*$ לכל השפות האחרות אי אפשר לעשות רדוקציה.



<u>הגדרה עבור שפות מאותו א"ב</u>: אם שתי השפות מעל אותו א"ב, נוכל להגדיר בצורה <u>פשוטה יותר</u>:

תהיינה $L_1, L_2 \subseteq \Sigma^*$, נאמר כי $L_1 \leq_P L_2$ (L_1 כלומר ניתנת לרדוקציה פולינומית ל-L_2) אם קיימת פונקציה $f: \Sigma^* \to \Sigma^*$ המקיימת:

1. f מלאה.
2. f ניתנת לחישוב **זמן פולינומי**.
3. f תקפה, כלומר $\forall x \in \Sigma^*: x \in L_1 \Leftrightarrow f(x) \in L_2$.

דוגמאות לרדוקציות פולינומיאליות

נתונות השפות הבאות:

$$Clique = \{< G, k > | G \text{ is a graph that contains a clique of size } k\}$$

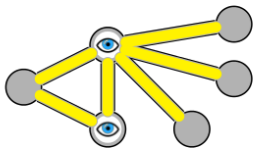
<u>תזכורת</u>: **קליקה,** $Clique$: בהינתן גרף $G = (V, E)$, קליקה ב-G היא תת קבוצה של קודקודים $S \subseteq V$ המקיימת שכל זוג קודקודים ב-S הם שכנים בגרף.

$$IS = \{< G, k > | G \text{ is a graph that contains an independent set of size } k\}$$

<u>תזכורת</u>: **קבוצה בלתי תלויה**,$Independent\ Set$: בהינתן גרף $G = (V, E)$, קבוצה בלתי תלויה ב-G היא תת קבוצה של קודקודים $S \subseteq V$ המקיימת שלא קיימות צלעות בגרף ששני הקודקודים החלים בהן שייכים ל-S. (במילים אחרות, לכל אחת מהצלעות ב-G יש לכל היותר קודקוד אחד ב-S).

$$VC = \{< G, k > | G \text{ is a graph that contains a vertex cover of size } k\}$$

<u>הגדרה</u>: **כיסוי קודקודים**, $Vertex\ Cover$: בהינתן גרף $G = (V, E)$, קבוצה $S \subseteq V$ היא קבוצת כיסוי קודקודים אם ורק אם כל צלע ב-G חלה בקודקוד מ-S (כלומר, אין צלע בגרף שאין לה קצה בקבוצה S).



<u>הערה</u>: חיפוש של $Clique$ או של VC לא מתבצע בזמן פולינומי, אבל המעבר בין שניהם (רדוקציה) הוא פולינומי ולכן נאמר שהם "קשים באותה המידה", כלומר שניהם יהיו באותה המחלקה.

<u>דוגמה 1</u>: איך עושים רדוקציה פולינומית $Clique \leq_P IS$?

<u>קלט</u>: $< G, k > \in Clique$ (כל זוג קודקודים ב-S הם שכנים בגרף G).

<u>פלט</u>: $< \bar{G}, k' > \in IS$ (לכל אחת מהצלעות ב-$\bar{G}$ יש <u>לכל היותר</u> קודקוד אחד ב-S).

<u>פונקצית הרדוקציה</u>: $f(< G, k >) =< \bar{G}, k >$.

<u>הסבר לפונקציה</u>: איך נתעסק בקליקה בשפה של IS? אם יש לנו בגרף G קליקה, אז כשנהפוך את כל הצלעות, אותה הקליקה בדיוק תהיה ב-$\bar{G}$ וה-k נשאר אותו הדבר כי זו אותה הקליקה בדיוק.

<u>הסבר כללי</u>: אנחנו לא חיפשו קליקה ב-IS, אנחנו נעזרנו ברדוקציה פולינומית שאומרת שאם יש קליקה ב-G אז בעזרת פונקצית הרדוקציה, יש לנו גם קליקה ב-$\bar{G}$. מאחר ולא חיפשנו (חיפוש שלם) אז זה פולינומי כי אנחנו בסה"כ עוברים על כל הצלעות בגרף והופכים אותם.

<u>פתרון דוגמה 1</u>:

<u>פונקצית הרדוקציה</u>: $f(< G, k >) =< \bar{G}, k >$.

<u>הרדוקציה פולינומית</u>: כדי לחשב את f יש לעבור על כל זוג קודקודים ב-G, אם הקודקודים שכנים ב-G, לא תהיה ביניהם צלע ב-$\bar{G}$ ואם הקודקודים לא שכנים ב-G תהיה ביניהם צלע ב-$\bar{G}$.

לכן הסיבוכיות היא $\binom{n}{2} = O(n^2)$, (כאשר n הוא מספר הקודקודים ב-G) <u>פולינומי</u>. (בהנחה שניתן לבדוק האם זוג קודקודים הם שכנים בזמן קבוע, <u>הכי קל</u>: לעבור על מטריצת השכנויות ולהפוך כל ביט).

<u>תקפות הרדוקציה</u>: התקפות נובעת מהלמה הבאה:

<u>למה</u>: עבור גרף $G = (V, E)$, קבוצת הקודקודים $S \in V(G)$ היא קליקה ב-G <u>אם"ם</u> S היא קבוצה ב"ת ב-$\bar{G}$.

<u>הוכחת הלמה</u>:

- כיוון א': נניח ש-S היא קליקה ב-G, אזי כל הצלעות האפשרויות של זוגות קודקודים מ-S קיימות ב-G, לכן ב-$\bar{G}$ כל הצלעות הללו לא קיימות, מכאן ש-S היא קבוצה בלתי תלויה ב-$\bar{G}$.
- כיוון ב': נניח ש-S היא קבוצה בלתי תלויה ב-$\bar{G}$, אזי כל הצלעות האפשרויות של זוגות קודקודים מ-S לא קיימות ב-$\bar{G}$, לכן ב-G כל הצלעות הללו קיימות, מכאן ש-S היא קליקה ב-G.

∎

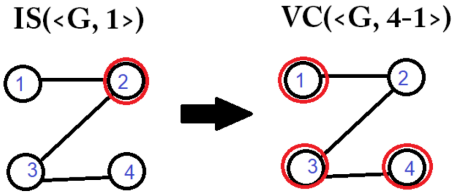
<u>דוגמה 2</u>: איך עושים רדוקציה פולינומית $IS \leq_P VC$?

<u>קלט</u>: $< G, k > \in IS$ (לכל צלע בגרף G, יש <u>לכל היותר</u> קודקוד אחד ב-S).

<u>פלט</u>: $< G, n - k > \in VC$ (לכל צלע בגרף G, <u>לפחות אחד</u> מקודקודיו שייך ל-S).

<u>פונקצית הרדוקציה</u>: $f(< G, k >) =< G, n - k >$.

<u>הסבר</u>: אם קיימת ב-G קבוצה בלתי תלויה בגודל k, אז קבוצת כיסוי הקודקודים ב-G היא בגודל $n - k$. ראו ציור:



<u>פתרון דוגמה 2</u>:

<u>פונקצית הרדוקציה</u>: $f(< G,k >) =< G,\ n - k >$.

<u>הרדוקציה פולינומית</u>: כדי לחשב את f, הרדוקציה רק מחשבת את מספר קודקודי הגרף $(n =)$ ומפחיתה מ-n ל-k. פעולות בזמן פולינומי.

<u>תקפות הרדוקציה</u>: תקפות הרדוקציה נובעת מהלמה הבאה:

<u>למה</u>: עבור גרף $G = (V, E)$ קבוצת קודקודים $S \subseteq V(G)$ היא בלתי תלויה ב-G אם"ם $V\backslash S$ היא כיסוי קודקודים של G.

<u>הוכחת הלמה</u>:

- כיוון א': נניח ש-S היא קבוצה בלתי תלויה ב-G, אזי אין צלעות המחברות <u>שני</u> קודקודים ב-S, כלומר, כל צלעות הגרף נוגעות בלכל היותר קודקוד אחד מ-S. אזי הקבוצה $V\backslash S$ נוגעת <u>בלפחות</u> קודקוד אחד מכל צלע בגרף, כלומר $V\backslash S$ היא כיסוי קודקודים.
- כיוון ב': נניח ש-$V\backslash S$ היא כיסוי קודקודים ב-G, אזי כל צלע ב-G היא בעלת <u>לפחות</u> קודקוד אחד בקבוצה $V\backslash S$. מכאן שאין צלעות שחלות על שני קודקודים מ-S, כלומר S היא קבוצה בלתי תלויה.

∎

<u>תרגיל בית</u>: הוכיחו שאם $L_1 \leq_P L_2$ אז $\overline{L_1} \leq_P \overline{L_2}$.

שייכות ל-P בעזרת רדוקציה פולינומית

<u>משפט הרדוקציה</u>: אם $L_1 \leq_P L_2$ ו-$L_2 \in P$ אזי גם $L_1 \in P$.

<u>הערה</u>: לא נשתמש במסקנה הבאה (מקביל למסקנה שהשתמשנו בחישוביות): אם $L_1 \leq_P L_2$ וגם $L_1 \notin P$, אז $L_2 \notin P$.

כלומר, לא נעסוק בשפות שהוכח עליהן שהן לא שייכות ל-P.

<u>הוכחת המשפט</u>: נניח כי $L_1 \leq_P L_2$ ו-$L_2 \in P$. תהי M_f מ"ט דטרמיניסטית פולינומית המחשבת את פונקציית הרדוקציה, ויהי p_1 הפולינום החוסם את זמן הריצה שלה. ותהי M_2 מ"ט דטרמיניסטית פולינומית המכריעה את L_2, ויהי p_2 חסם זמן הריצה שלה.

נתאר מכונה M_1 דטרמיניסטית פולינומית המכריעה את L_1:

M_1 על הקלט x תפעל:

1. מחשבת את $f(x)$. *(סיבוכיות: $O(p_1(|x|))$).*
2. מריצה את M_2 על $f(x)$ <u>ועונה כמוה</u>. *(סיבוכיות: $(p_2(p_1(|x|)))^2$).*

<u>הוכחת נכונות הבניה</u>: נכונות המכונה נובעת ישירות מהתקפות של פונקצית הרדוקציה.

<u>סיבוכיות זמן ריצה</u>: זמן הריצה של המכונה M_1 הוא $O\left(p_1(|x|) + (p_2(p_1(|x|)))^2\right)$ הוא *פולינומי*.

∎

הערה לסיבוכיות של ההוכחה: מכונה המריצה מכונה אחרת עולה לנו יותר זמן, כי יש לנו סרט נוסף שכותבים בו את הקונפיגורציה של המכונה כדי שנדע לסמלץ אותה ולמדנו שהסרט הנוסף הזה הוא עד גודל ריבועי.
לכן תמיד שיש מכונה שמריצה מכונה נוספת אנחנו נוסיף לסיבוכיות שלה "בריבוע".

משפטים נוספים

משפט 1: אם $L_1 \leq_P L_2$ ו-$L_2 \in NP$ אז גם $L_1 \in NP$.

תרגיל 1: הוכיחו את משפט 1.

פתרון תרגיל 1: **להשלים**.

הערה: גם כאן לא נשתמש במסקנה מן המשפט כדי להוכיח ששפות אינן שייכות ל-NP.
לא נעסוק בשפות שהוכח עליהן שהן לא שייכות ל-NP.

משפט 2: אם $L_1 \in P$ ו-L_2 היא שפה כלשהי ששונה מ-$\emptyset$ וגם מ-Σ^*, אז $L_1 \leq_P L_2$.

תרגיל 2: הוכיחו את משפט 2.

פתרון תרגיל 2: **להשלים**.

משפט 3: היחס $\leq_P$ טרנזיטיבי.

תרגיל 3: הוכיחו שהיחס $\leq_P$ הוא טרנזיטיבי. - אם $L_1 \leq_P L_2$ וגם $L_2 \leq_P L_3$ אז $L_1 \leq_P L_3$.

פתרון תרגיל 3: **להשלים**.

תרגיל 4: האם היחס $\leq_P$ הוא רפלקסיבי? - הוכיחו.

פתרון תרגיל 4: הוראות לפתרון - האם $A \leq_P B$ אז גם $B \leq_P A$? - לא ידוע אם זה רפלקסיבי. כי אם $A \in P$ וגם $B \in NP$, מאחר ולא ידוע האם $NP \subseteq P$ אז לא ידוע האם רפלקסיבי. **להשלים**.

תרגיל 5: האם היחס $\leq_P$ הוא סימטרי? - הוכיחו.

פתרון תרגיל 5: הוראות לפתרון - מהי פונקצית רדוקציה פולינומית משפה לעצמה? - כלום, זה פונק' הזהות ולכן זה סימטרי. **להשלים**.

הערה: אינטואיטיבית, אם $L_1 \leq_P L_2$, אז L_2 קשה לפחות כמו L_1 (במובן של קיום אלגוריתם מהיר לשפה).
אם יש אלגוריתם מהיר לבדיקת השייכות ל-L_2, אז אפשר לבנות בעזרתו (ובעזרת הרדוקציה הפולינומית) אלגוריתם מהיר לבדיקת השייכות ל-L_1.

השפות הקלות ב-NP

שאלה: מי הן השפות הקלות ביותר במחלקה NP (ללא השפות הטריוויאליות)?

- לפי הסיווג של קל/קשה בעזרת היחס $\leq_P$.

תשובה: אלה השפות (הלא טריוויאליות) ב-P.

הסבר: אם $L_1 \in P$ אז $L_1 \leq_P L_2$ לכל L_2 במחלקה. כל L_2 במחלקה קשה לפחות כמו L_1 (L_1 הקלה ביותר).

שפות NP שלמות

מחלקה NP-שלמות

הצבענו על הדמיון שקיים בין המחלקה RE והמחלקה NP, ובין המחלקה R והמחלקה P.
בהרצאה הזו נרחיב את הדמיון הזה גם למחלקה $RE - Complete$, ונלמד על המקבילה שלה - שפות NP-שלמות או $NP - Complete$. (אולי אחת ההגדרות הכי חשובות במדעי המחשב).
<u>הסבר תכל'ס</u>: NP-שלם (לעיתים נסמן ב-NPC) אומר שלא ידוע על אלגוריתם (מכונה) **יעיל** (פולינומי) לשפה, כלומר אם נצליח להראות רדוקציה משפה כלשהי לשפה $NP - Complete$ (או להפך) אז אנחנו הוכחנו **שלא קיים** אלגוריתם יעיל (פולינומי). אמרנו שלשפה ב-NP קיימת מ"ט א"ד פולינומית שמנחשת "עד" ואז בודקת (validation) עליו. המחשב לא יודע לנחש (הוא דטרמיניסטי) אז כל עוד נקבל ששפה היא NP-שלמה, נדע שלא צריך לבזבז זמן יקר בלחפש עבורה אלגוריתם פולינומי (כי אין כזה).

תזכורת: RE-Complete

שפה L_2 היא שלמה ב-RE אם:

1. $L_2 \in RE$ (שייכת למחלקה).
2. לכל שפה $L_1 \in RE$ מתקיים ש: $L_1 \leq L_2$ (קשה ביותר במחלקה).

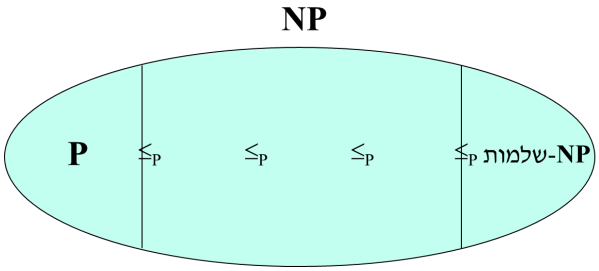
חשוב: השפות השלמות ב-RE הן הקשות ביותר במחלקה RE.
<u>שפות NP קשות (NPH)</u>: שפה $L_2 \in NPH$ קשה אם לכל $L_1 \in NP$ קיימת רדוקציה פולינומית $L_1 \leq_P L_2$.

הגדרה: שפות NP-שלמות

שפה L_2 היא שלמה ב-NP אם:

1. $L_2 \in NP$ (שייכת למחלקה).
2. לכל שפה $L_1 \in NP$ מתקיים ש: $L_1 \leq_P L_2$ (קשה ביותר במחלקה).

מדובר על שפות השייכות למחלקה, ולכל שפה במחלקה יש רדוקציה בזמן פולינומיאלי אליהן.
חשוב: השפות השלמות ב-NP הן הקשות לפחות כמו כל שפה אחרת במחלקה NP.



איך נכריע האם P=NP

משפט: אם יש שפה NPC (שלמה) ששייכת ל-P, אז $P = NP$.

תרגיל: הוכיחו את המשפט.

מסקנה:

- אם רוצים להוכיח ש: $P = NP$, די להראות מ"ט דטרמיניסטית בזמן ריצה פולינומי לאחת השפות השלמות ב-NP.
- אם רוצים להוכיח ש: $P \neq NP$, אפשר להוכיח על שפה $L \in NP$ שאין לה מ"ט דטרמיניסטית בזמן ריצה פולינומי. סביר ש-L תהיה NPC, (**למה?**).

האם יש שפות שלמות במחלקה NPC?

שאלה: נותרה השאלה האם יש בכלל שפות NP-שלמות?. כלומר, האם יש שפות ב-NP כך שלכל שפה ב-NP יש רדוקציה בזמן פולינומי אליהן?.

תשובה: יש שפות כאלה. יש במחלקה NP שפות שלמות. (שפות קשות ביותר מחלקה).

השפה הראשונה שהוכח עליה שהיא NPC היא **שפת הפסוקים הספיקים בתחשיב הפסוקים**.

תזכורת מהקורס בלוגיקה: פסוק בתחשיב הפסוקים בנוי ממשתנים פסוקיים (אטומים) ומְקַשְׁרִים.

- המשתנים הפסוקיים (אטומים) - יכולים לקבל ערך $True$ (או 1), או $False$ (או 0).
- ערך האמת - של הפסוק $True\ or\ False$ נקבע לפי *טבלאות האמת* של הקשרים.
- סוגי קשרים - אנחנו נסתפק בשלושת הקשרים הבאים: $\neg$, $\wedge$, $\vee$.
- שם הנוסחה - נכתוב $\phi(x_1, x_2, ..., x_n)$.

פסוקים ספיקים

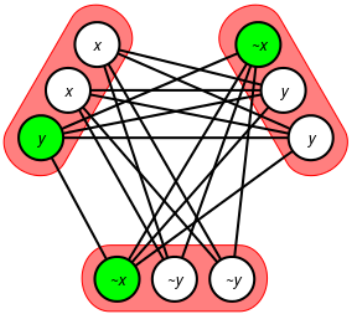
הגדרה: פסוק נקרא **ספיק** אם יש לפחות השמה אחת של ערכי אמת למשתנים של הפסוק שבה ערך האמת של הפסוק הוא $true$.

תרגיל: הפסוק $\phi(x, y) = (x \vee \neg y) \wedge (\neg x \vee y)$ הוא פסוק מעל המשתנים x ו-y. האם הוא ספיק?

תשובה: בהחלט. הנוסחה מקבילה לשער $XNOR$.

בעיית הספיקות - השפה SAT

מבוא



בעיית הספיקות בתחשיב הפסוקים (בקיצור: SAT - קיצור של המילה האנגלית $Satisfiability$, ספיקות) היא בעיית הכרעה הנחקרת במסגרת תורת הסיבוכיות. בעיה זו הייתה הבעיה הראשונה עליה הוכח כי היא NP-שלמה (הוכחה זו היא *משפט קוק-לוין*), משמע אם קיים לה פתרון אלגוריתמי הרץ בזמן פולינומי אזי קיים פתרון כזה לכל בעיה ב-NP (מחלקת כל הבעיות שעבורן קיים מוודא בזמן פולינומי). בעיה זו משמשת בהוכחות רבות עבור בעיות NP-שלמות אחרות.

הגדרה - השפה SAT

השפה SAT היא שפת הפסוקים הספיקים בתחשיב הפסוקים:

$$SAT = \{< \phi > \mid \phi \; is\; a\; satisfiable\; Boolean\; formula\}$$

משפט Cook-Levin

<u>משפט</u>: $SAT \in NPC$. כלומר, השפה SAT היא NP-שלמה.

<u>רעיון ההוכחה</u>: צריך להוכיח שני דברים:

1. $SAT \in NP$.
2. לכל שפה $L \in NP$ מתקיים ש: $L \leq_P SAT$.

משפט Cook-Levin, חלק א': הוכחה $SAT \in NP$.

להלן N_L מ"ט א"ד פולינומית:

1. נחש השמה למשתנים $x_1,...,x_n$.
2. הצב את ערך הניחוש, וודא שהפסוק מקבל $True$.

<u>הוכחת נכונות</u>:

- $< \phi > \in SAT$ ← קיימת השמה מספקת לביטוי ϕ ← המכונה N_L תנחש את ההשמה הנכונה ← המכונה תענה $True$ ← $< \phi > \in L(N_L)$.
- $< \phi > \notin SAT$ ← לא קיימת השמה מספקת לביטוי ϕ ← לכל ניחוש של N_L התשובה תהיה $False$ ← $< \phi > \notin L(N_L)$.

משפט Cook-Levin, חלק ב': הוכחת הרדוקציה הפולינומית

צריך להראות **שלכל שפה** $L \in NP$ מתקיים $L \leq_p SAT$.

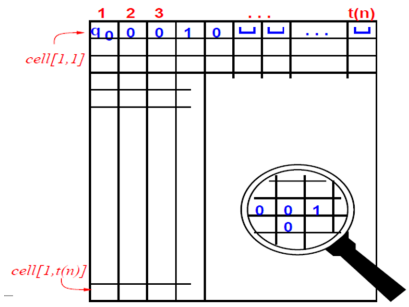
<u>מה אנחנו יודעים על L?</u>: לא הרבה. אנחנו יודעים שהיא ב-NP. כלומר, יש לה מכונה לא דטרמיניסטית מכריעה N שרצה בזמן $O(n^k)$ פולינומי ל-k טבעי כלשהו.

<u>איך תיראה הרדוקציה?</u>: הרדוקציה תקבל כקלט מילה w מעל האלפבית של L.
הרדוקציה תבנה (בזמן פולינומיאלי ב-$|w|$) פסוק בתחשיב הפסוקים שיהיה ספיק *אם"ם* $w \in L$.
כלומר, הפסוק ייבנה כך שהוא יהיה ספיק אם"ם יש ל-N חישוב מקבל על w.

<u>תמונה של מסלול חישוב</u>: נניח אם כן שיש ל-$L \in NP$ מכונה א"ד מכריעה N שזמן הריצה שלה על מילה w חסום על-ידי n^k כאשר ($n = |w|$) ל-k טבעי כלשהו (קבוע).
בתוך n^k צעדים N יכולה להגיע לכל היותר לריבוע ה-n^{k+1} על הסרט שלה (מניחים של-N יש סרט יחיד).

<u>מטריצת tableau</u>: נשתמש בטבלה הזאת (קונספטואלית) כדי לייצר פסוק (לא כחלק מהבניה).
תמונה של ריצת N על w באחד ממסלולי החישוב היא מטריצה ($tableau$) מסדר $(n^k + 1) \times (n^k + 2)$ -

- השורה הראשונה מתאימה לקונפיגורציה ההתחלתית.
- השורה ה-i מתאימה לקונפיגורציה ה-$i - 1$ בחישוב.
- השורה האחרונה (ה-$n^k + 1$) מתאימה לקונפיגורציה ה-n^k.



<u>הנחות</u>: לשם הנוחות, נניח שכל קונפיגורציה מתחילה ומסתיימת ב-#.

1. בכל העמודה הראשונה ובכל העמודה האחרונה במטריצה מופיע הסמל #.
2. נרחיב את המטריצה לסדר $(n^k + 4) \times (n^k + 1)$.
3. לשם הנוחות של האינדקסים בהמשך נניח שהיא מסדר $(n^k + 1) \times (n^k + 1)$ גם נניח שאם מגיעים לקונפיגורציה עוצרת (קבלה או דחייה), מעתיקים אותה לשורות שתחתיה במטריצה.

כך נבטיח שתמיד המטריצה היא מסדר $(n^k + 1) \times (n^k + 1)$.

<u>מטריצה מקבלת</u>: המטריצה תקרא מקבלת, אם אחת השורות שלה היא קונפיגורציה מקבלת.
כלומר, המצב בקונפיגורציה הוא q_{accept}. כל מטריצה מקבלת של N על w מתאימה לחישוב מקבל של N על w. לכן, השאלה האם N מקבלת את w שקולה לשאלה "האם יש מטריצה מקבלת של N על w".

אנו מעוניינים ברדוקציה פולינומית עבור $\forall L \in NP,\ L \leq_P SAT$. ולכן, לכל מילת קלט נרצה לבנות פסוק.

1. תחילה בהינתן מילה x נבנה פסוק שיסתפק אמ"מ יש מסלול מקבל בעץ החישוב $N(x)$ (כלומר מטריצה מקבלת) ונסביר נכונות (כלומר את תקפות הרדוקציה).
2. לאחר מכן נוכיח סיבוכיות אורך הפסוק.

<u>מה תעשה הרדוקציה?</u>: הרדוקציה תבנה מ-w פסוק שערכו יהיה $true$ אם"ם יש מטריצה מקבלת של N על w
<u>איך יראה הפסוק הזה?</u>: הסמלים שיכולים להופיע במטריצה הם סמלים מן הקבוצה $C = Q \cup \Gamma \cup \{\#\}$ או מיקום של הראש. שימו לב שגודל הקבוצה C לא תלוי ב-w (אלא רק ב-N), ולכן נחשב **קבוע** בחישוב זמן הריצה.

<u>הפסוק שבונה הרדוקציה</u>: לכל $1 \leq i,j \leq n^k + 1$ ולכל סימן $s \in C$ נגדיר משתנה $x_{i,j,s}$.
הכוונה היא ש: $x_{i,j,s} = true$ אם"ם $tableau[i][j] == s$.
מספר המשתנים הוא $O(|i| \cdot |j| \cdot |c|) = O(n^k \cdot n^k \cdot 1) = O(n^{2k})$.

הרדוקציה תבנה מ-w את הפסוק ϕ שהוא קוניונקציה (AND) של ארבעה פסוקים:

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

נפרט על כל אחד מן הרכיבים של הפסוק:

ϕ_{cell}	וידוא שבכל תא במטריצה יכול להיות בדיוק תו אחד.
ϕ_{start}	וידוא שהשורה הראשונה מכילה את הקונפיגורציה ההתחלתית המתאימה שכן אם יש לנו מטריצה מקבלת אבל היא לא התחילה בקונפיגורציה ההתחלתית היא לא יכולה לאשר את קבלת המילה.
ϕ_{move}	תוודא שהמעברים בין הקונפיגורציות חוקיים לפי פונקציית המעברים δ.
ϕ_{accept}	שהקונפיגורציה הסופית תכיל מצב מקבל, אמרנו הרי שאם יש פחות מ-n^k קונפיגורציות פשוט נשכפל את הקונפיגורציה האחרונה מספר פעמים, ולכן יספיק לוודא מצב מקבל רק בשורה האחרונה.

הפסוק – ϕ_{cell} המשמעות של ϕ_{cell} היא שבכל תא במטריצה יש סמל אחד ויחיד מתוך C.
לכל i,j יש s אחד ויחיד כך ש: $x_{i,j,s}$ הוא $true$. בניסוח של פסוק נכתוב:

$$\phi_{cell} = \wedge_{1 \leq i,j \leq n^k+1} \left[\left(\vee_{s \in C} x_{i,j,s} \right) \wedge \left(\wedge_{s,t \in C,\ s \neq t} \left(\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right) \right]$$

<u>הביטוי השמאלי</u>: לוודא שאחד מהם הוא T.
<u>הביטוי הימני</u>: לוודא שכל האחרים הם F.

סיבוכיות הפסוק:

- עבור תא יחיד: $O(|\Gamma \cup Q|^2) + O(\binom{|\Gamma \cup Q|}{2}) = O(|\Gamma \cup Q|^2)$.
- עבור כל התאים במטריצה: $O(n^{2k} \cdot |\Gamma \cup Q|^2)$.

הפסוק ϕ_{start} – המשמעות של ϕ_{start} היא שהשורה הראשונה של המטריצה היא הקונפיגורציה ההתחלתית של N על w. בכל תא בשורה הראשונה במטריצה מופיע הסמל הנכון של הקונפיגורציה ההתחלתית.
בניסוח של פסוק:

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\bar{b}} \wedge \ldots \wedge x_{1,n^k,\bar{b}} \wedge x_{1,n^k+1,\#}$$

סיבוכיות הפסוק: **להשלים**

הפסוק ϕ_{accept} – המשמעות של ϕ_{accept} היא שיש במטריצה קונפיגורציה מקבלת.
הדרישה היא שהסמל q_{accept} מופיע בשורה האחרונה:

$$\phi_{accept} = \bigvee_{1 \le j \le n^k+1} x_{n^k+1,\, j\, ,\, q_{accept}}$$

סיבוכיות הפסוק: $O(n^k)$.

הפסוק ϕ_{move} – המשמעות של ϕ_{move} היא לוודא שכל שורה (קונפיגורציה) מאפשרת לעבור לשורה הבאה (קונפיגורציה עוקבת) דרך פונקציית המעברים של N.

הנקודה החשובה היא: ההבדל בין שתי קונפיגורציות עוקבות מתבטאת בלכל היותר בשלושה תאים רצופים (מיקום הראש, והתאים שלימינו ולשמאלו) וכל השאר זהה. לכן די לוודא שכל "חלון" מסדר 3×2 הוא חוקי לפי N.

<u>חלונות חוקיים ולא חוקיים</u>: לדוגמה נניח שב-N מתקיים:

$$\delta(q_1, a) = \{(q_1, b, R)\} \quad , \quad \delta(q_1, b) = \{(q_2, c, L),\ (q_2, a, R)\}$$

חלונות לא חוקיים

b	q_1	b
q_1	b	q_2

a	q_1	b
q_1	a	a

a	b	a
a	a	a

חלונות חוקיים

a	a	q_1
a	a	b

b	b	b
c	b	b

a	q_1	b
a	a	q_2

a	b	a
a	b	q_2

a	q_1	b
q_2	a	c

#	b	a
#	b	a

לכל חלון מסדר 3×2 יהיה ב-ϕ_{move} תת-פסוק שיאמר שששת הסמלים בחלון מהווים חלון חוקי:

$$\phi_{move} = \bigwedge_{\substack{1 \le i < n^k + 1 \\ 1 < j < n^k + 1}} (the\ (i,j)\ window\ is\ legal)$$

כאשר הקביעה שהחלון הוא חוקי היא הפסוק הבא בזמן קבוע $O(1)$ לכל בדיקת חלון 3×2:

$$\bigvee_{a_1,\ldots,a_6\ is\ a\ legal\ window} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6})$$

(פסוקים כאלה מופיעים לכל i ולכל j ולכל חלון חוקי) שימו לב שמספר החלונות החוקיים איננו תלוי כלל ב-w (אלא רק במכונה N), ולכן הוא ייחשב <u>לקבוע</u> בחישוב זמן הריצה של הרדוקציה.

<u>סיבוכיות הפסוק</u>: $O(6 \cdot (n^k + 1)(n^k + 1)) = O(n^{2k})$.

<u>טענה</u>: אם השורה הראשונה במטריצה היא הקונפיגורציה ההתחלתית של N על w, וכל חלון מסדר 3×2 במטריצה הוא חלון חוקי לפי N, אז כל שורה בטבלה היא קונפיגורציה עוקבת לקונפיגורציה של השורה שלפניה, בְּחישוב של N על w. **להשלים?**

<u>הדרכה</u>: לכל סמל בקונפיגורציה שאיננו #, התבוננו בחלון שבו הסמל הזה הוא הסמל האמצעי בשורה העליונה בחלון.

<u>הרדוקציה תקפה</u>: **להשלים?**

<u>תרגיל</u>: הוכיחו: $\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$ ספיק אם"ם $w \in L$.

תיארנו את הרדוקציה, והוכחנו שהיא תקפה (הוכחנו בבית). כעת נעבור לחישוב זמן הריצה של הרדוקציה, ונוכיח שהוא פולינומי בגודל הקלט. זה יסיים את הוכחת משפט $Cook - Levin$.

<u>הגודל של הפסוק ϕ</u>: נחשב את הגודל של ϕ:

- מספר המשתנים הוא $O(n^{2k})$.
- הגודל של ϕ_{cell} הוא $O(n^{2k})$.
- הגודל של ϕ_{start} הוא $O(n^{k})$.
- הגודל של ϕ_{move} הוא $O(n^{2k})$.
- הגודל של ϕ_{accept} הוא $O(n^{k})$.

מסקנה: הגודל של ϕ הוא $O(n^{2k})$.

<u>זמן הריצה של הרדוקציה</u>:

אפשר לחשב את ϕ בזמן פולינומיאלי ב-$|w|$:

- ϕ_{move}, ϕ_{cell}, ϕ_{accept} לא תלויים ב-w עצמה, אלא רק ב-$|w|$ (כדי לקבוע את התחום של האינדקסים i ו-j).
- הרכיבים של הפסוקים הללו חוזרים על עצמם, ולכן אפשר לחשב אותם בזמן פולינומיאלי.
- את ϕ_{start} אפשר לחשב מתוך w בזמן פולינומיאלי.

<u>סיום ההוכחה המגעילה הזאת</u>: בזה תמה הוכחת משפט $Cook - Levin$ ומכאן SAT היא NP-שלמה!

צורה קוניונקטיבית נורמלית

<u>תזכורת</u>: פסוק בתחשיב הפסוקים הוא בצורה קוניונקטיבית נורמלית (CNF), אם:

1. הפסוק בנוי מקבוצה של פסוקיות שביניהן יש הקשר $\wedge$
2. כל פסוקית היא קבוצה של ליטרלים שביניהם יש הקשר $\vee$
3. כל ליטרל הוא משתנה או שלילה של משתנה

<u>דוגמה</u>: $(x \vee y \vee \neg z) \wedge (x \vee z) \wedge (\neg y \vee \neg z)$.

<u>תזכורת</u>: לכל פסוק יש <u>פסוק שקול</u> ב-CNF.

השפה CNF-SAT

תרגיל: הראו שכל חלקי הפסוק ϕ בהוכחת משפט $Cook - Levin$, פרט ל-ϕ_{move}, הם בצורת CNF.
אפשר להעביר גם את ϕ_{move} לצורת CNF. זה יגדיל את הפסוק, אך מכיוון שהגודל של כל תת-פסוק של חלון ב-ϕ_{move} תלוי רק במכונה N (ולא ב-w), אפשר להתייחס להגדלה הזו כאל כפל בקבוע.

מסקנה: גם השפה $CNF - SAT$ היא NPC:

$$CNF - SAT = \{< \phi > \mid \phi \; is \; a \; satisfiable \; CNF \; Boolean \; formula\}$$

השפה DNF-SAT

תזכורת: פסוק בתחשיב הפסוקים הוא בצורה דיסיונקטיבית נורמלית (DNF), אם:

1. הפסוק בנוי מקבוצה של פסוקיות שביניהן יש הקשר $\wedge$
2. כל פסוקית היא קבוצה של ליטרלים שביניהם יש הקשר $\vee$
3. כל ליטרל הוא משתנה או שלילה של משתנה

$$DNF - SAT = \{< \phi > \mid \phi \; is \; a \; satisfiable \; DNF \; Boolean \; formula\}$$

תרגיל: הוכיחו: $DNF - SAT \in P$. שימו לב שבין הפסוקיות יש הקשר $\vee$.

שאלה: האם גם $CNF - SAT \in P$? למה האלגוריתם הבא ל-$CNF - SAT$ לא מוכיח שגם $CNF - SAT$ שייכת ל-P?

- "על קלט $< \phi >$ כאשר ϕ פסוק בצורת CNF:
  1. העבר את ϕ לפסוק שקול 'ϕ בצורת DNF.
  2. בדוק האם 'ϕ ספיק. אם כן, קבל. אחרת, דחה."

תשובה: המעבר מ-CNF ל-DNF עלול ליצור פסוק 'ϕ שגודלו אקספוננציאלי בגודל הפסוק ϕ.

שקילות CNF ו-DNF

לדוגמה: $\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_4) \wedge \ldots \wedge (\neg x_{n-1} \vee \neg x_n)$.
כיצד ממירים את זה לצורת DNF?

לעיון בדוגמה "קצרה" בקישור.

סיכום הפרק

<u>רדוקציה פולינומית</u>: בהינתן שתי שפות L_1, L_2, נאמר ש: L_1 ניתנת לרדוקציה פולינומית ל-L_2 ונסמן $L_1 \leq_P L_2$ אם קיימת פונקציה f כך ש:

1. f ניתנת לחישוב בזמן פולינומי.
2. f תקפה, כלומר $x \in L_1 \Leftrightarrow f(x) \in L_2$.

<u>שפות NP-שלמות</u>: שפה L היא NP-שלמה (או נסמן NPC) אם:

1. $L \in NP$.
2. $L \in NPH$ כלומר, לכל $L' \in NP$, קיימת רדוקציה פולינומית $L' \leq_P L$.

<u>השפה SAT</u>: היא שפת הפסוקים הספיקים בתחשיב הפסוקים:

$$SAT = \{< \phi > \mid \phi \; is\; a\; satisfiable\; Boolean\; formula\}$$

הוכחנו בהרצאה בעזרת משפט קוק-לוין ש: $SAT \in NPC$.

תרגיל 1 - תרגול 6

נתונה השפה הבאה: $SAT_2 = \{< \phi > \mid \exists\; two\; satisfying\; assignments\; to\; \phi\}$.

<u>במילים</u>: שפת כל הפסוקיות ϕ שיש להן 2 השמות מספקות (שונות).

הוכיחו: $SAT_2 \in NPC$.

<u>פתרון</u>: כדי להוכיח ש: $SAT_2 \in NPC$, צריך להוכיח כי $SAT_2 \in NP$ וגם $SAT_2 \in NPH$.

<u>פתרון - חלק א'</u>: נוכיח $SAT_2 \in NP$. צריך לבנות מכונה א"ד N_2 בזמן פולינומי שמכריעה את השפה SAT_2 ולהוכיח נכונות וסיבוכיות. אנחנו ננחש שתי השמות ונבדוק שהן מספקות.

המכונה N_2 על קלט $< \phi >$ תפעל:

1. נחש השמה π_1 ונחש השמה נוספת π_2 כאשר $\pi_1 \neq \pi_2$.
2. בדוק ש-π_1 השמה מספקת, כלומר בדוק ש: $\phi(\pi_1) = true$.
3. בדוק ש-π_2 השמה מספקת, כלומר בדוק ש: $\phi(\pi_2) = true$.
4. אם π_1 וגם π_2 השמות מספקות עבור ϕ - <u>קבל</u>. אחרת, <u>דחה</u>.

<u>סיבוכיות זמן הריצה</u>: נסמן ב-n את מספר המשתנים ב-ϕ.

1. שלב 1: כגודל הניחוש $O(|\pi_1|) = O(n)$ וגם $O(|\pi_2|) = O(n)$, סה"כ $O(2n) = O(n)$.
2. שלב 2-3: בבדיקת ההשמה נבצע הצבה בפסוק, זה יקח לנו כגודל ϕ, כלומר $O(|\phi|)$.
3. שלב 4: ביצוע בדיקה בזמן קבוע $O(1)$.

סך הכל $O(|\phi| + n)$ - פולינומי.

הוכחת נכונות בניית המכונה: נוכיח כי $SAT_2 = L(N_2)$.

- $<\phi> \in SAT_2 \leftarrow$ קיימות שני השמות מספקות שונות π_1, π_2 עבור $\phi \leftarrow$ קיים ניחוש של π_1 ושל π_2 המובילים למצב מקבל $\leftarrow <\phi> \in L(N_2)$.
- $<\phi> \notin SAT_2 \leftarrow$ לכל השמות π_1, π_2 הם לא מספקות את $\phi \leftarrow$ לכל ניחוש N_2 דוחה $\leftarrow <\phi> \notin L(N_2)$.

<u>פתרון - חלק ב'</u>: נוכיח כי $SAT_2 \in NPH$. כלומר שלכל $L' \in NP$ מתקיים $L' \leq_P SAT_2$.

<u>מבוא</u>: בהרצאה למדנו כי $SAT \in NPC$, כלומר ש: SAT שפה NP-שלמה. בנוסף, למדנו שבשפות שלמות ניתן בעזרת רדוקציה להוכיח על שפות שלמות אחרות. כלומר, אם כל שפה $L' \in NP$ ניתנת לרדוקציה עם SAT, אז כאשר נוכיח ש: $SAT \leq_P SAT_2$ גם $SAT_2 \in NPH$.

- <u>תיאור הרדוקציה</u>: הוכחנו כי $SAT \in NPC$, לכן נוכיח ברדוקציה פולינומית $SAT \leq_P SAT_2$.
  - פונקצית הרדוקציה: $f(\phi) = \phi'$ כאשר $\phi' = \phi \wedge (y \vee \bar{y})$.
  - אין צורך לתאר את האלגוריתם מכיוון שהפונקציה מתארת היטב את הפעולה הנדרשת.
- <u>הוכחת נכונות פונקצית הרדוקציה הפולינומית</u>:
  - הפונקציה f ניתנת לחישוב בזמן פולינומי, כי צריך בסה"כ להוסיף מחרוזת קבועה.
  - <u>תקפות הפונקציה</u>: צריך להוכיח $<\phi> \in SAT \Leftrightarrow <\phi'> \in SAT_2$.
    - $<\phi> \in SAT \leftarrow$ קיימת השמה מספקת π לפסוק $\phi \leftarrow$ ההשמות $\pi_1 = \pi \cdot 0$ וגם $\pi_2 = \pi \cdot 1$ מספקות את ϕ' כיוון שקיימת בו פסוקית טאוטולוגית (תמיד נכונה) $\leftarrow$ יש ל-ϕ' לפחות שתי השמות מספקות $\leftarrow <\phi'> \in SAT_2$.
    - $<\phi'> \in SAT_2 \leftarrow$ קיימות שני השמות מספקות שונות π_1, π_2 לפסוק $\phi' \leftarrow$ נבחר בה"כ את ההשמה המספקת $\pi_1 \leftarrow$ ההשמה π_1 בפרט מספקת את הפסוק $\phi \leftarrow <\phi> \in SAT$.

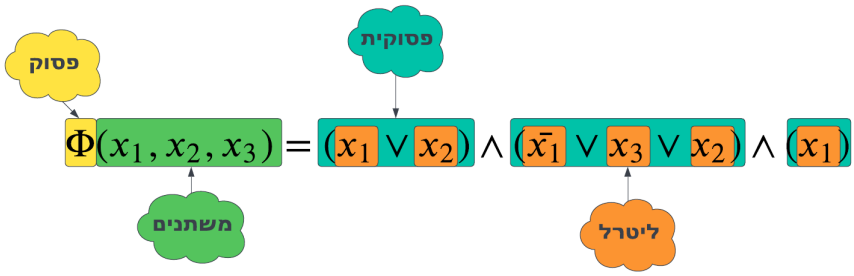
<u>סיכום פתרון</u>: הוכחנו $SAT_2 \in NP$ וגם הראנו ש: $SAT_2 \in NPH$ בעזרת רדוקציה פולינומית $SAT \leq_P SAT_2$ ולכן $SAT_2 \in NPC$. כלומר, SAT_2 היא שפה NP-שלמה - מ.ש.ל.

∎

תרגיל 2 - שרשרת רדוקציות (תרגיל גדול) - תרגול 6

<u>תזכורת</u>:

- פסוק CNF הוא ספיק אם קיימת השמה שנותנת לו ערך T.
- שפת $CNF - SAT$ היא שפת כל פסוקי ה-CNF הספיקים.
- אם יש לנו פסוק CNF שבו כל פסוקית מכילה <u>בדיוק</u> k ליטרלים, אומרים שזהו פסוק $k - CNF$.
- השפה $k - CNF - SAT$ היא אוסף פסוקי ה-$k - CNF$ הספיקים.



<u>שרשרת הרדוקציות</u>: בתרגיל זה אנחנו נוכיח שייכות של שפות למחלקה NPC בעזרת שרשרת רדוקציות.
היות ו-SAT היא שפה NP-שלמה (לפי משפט קוק לוין) אז גם שאר השפות הבאות יהיו NP-שלמות.
מדובר בתרגיל גדול, השפות שנוכיח כאן יהיו כלים לפתירת בעיות במבחנים ומטלות.
כל רדוקציה ($\leq_P$) הוא תרגיל בפני עצמו (יש 6 כאלה). עבור כל שפה בשרשרת הנתונה צריך להוכיח שהיא שייכת למחלקה NP ונעשה זאת ע"י בניית מ"ט א"ד פולינומית + הוכחת בנייה וסיבוכיות זמן ריצה.
לאחר שנוכיח שכל השפות $L \in NP$ אז נפתור כל רדוקציה פולינומית נתונה בנפרד (משמאל לימין).
נוכיח בשלבים את שרשרת הרדוקציות הבאה:

$$CNF - SAT \leq_P 3 - CNF - SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>הגדרת שפות</u>:

$$DHL = \{< G > \mid G\ is\ a\ directed\ graph\ with\ a\ Hamiltonian\ path\}$$
$$HL = \{< G > \mid G\ is\ an\ undirected\ graph\ with\ a\ Hamiltonian\ path\}$$
$$HC = \{< G > \mid G\ is\ an\ undirected\ graph\ with\ a\ Hamiltonian\ cycle\}$$
$$DHC = \{< G > \mid G\ is\ a\ directed\ graph\ with\ a\ Hamiltonian\ cycle\}$$
$$VC = \{< G, k > \mid G\ is\ a\ graph\ that\ contains\ a\ vertex\ cover\ of\ size\ k\}$$
$$3 - CNF - SAT = \{< \phi > \mid \phi\ is\ a\ satisfiable\ 3 - CNF\ Boolean\ formula\}$$
$$CNF - SAT = \{< \phi > \mid \phi\ is\ a\ satisfiable\ CNF\ Boolean\ formula\}$$

$$CNF-SAT \leq_P 3-CNF-SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>רדוקציה 1</u>: $HL \leq_P DHL$. תחילה צריך להוכיח כי $DHL \in NP$ וגם $HL \in NP$. עבור $HL \in NP$ הוכחנו כבר בהרצאה קודמת, קראנו לשפה $HL = HAMPATH$. צריך להוכיח $DHL \in NP$.

<u>בניית מכונה ל-DHL</u>: צריך להוכיח $DHL \in NP$. לכן, נבנה N מ"ט א"ד פולינומית שתכריע את השפה DHL.

המכונה N על קלט $< G >$ תבצע:

1. נחש מסלול מצומת s לצומת t.
2. בדוק שכל צלע e במסלול שייכת ל-$E(G)$ $\in e$.
   a. אם $e \notin E(G)$ - <u>דחה</u>.
3. לכל צומת $v_i \in V(G)$:
   a. אם ביקרנו ב-v_i יותר מפעם אחת - <u>דחה</u>.
   b. אם לא ביקרנו ב-v_i בכלל - <u>דחה</u>.
4. <u>קבל</u>.

<u>סיבוכיות זמן הריצה</u>: נסמן ב-d את אורך המסלול.

- שלב 1 - כגודל הניחוש: $O(d)$.
- שלב 2 - בדיקת כל צלע על המסלול $O(d \cdot |E|) = O(|E|^2)$.
- שלב 2a - בדיקה ספציפית - $O(1)$.
- שלב 3 - מעבר על כל צומת $O(|V|)$.
- שלב 3a,b - בדיקה ספציפית - $O(1)$.

סה"כ זמן הריצה הוא $O(|V| + |E|^2)$ - <u>פולינומי</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $DHL = L(N)$.

- $< G > \in DHL \leftarrow$ קיים מסלול המילטוני מכוון בגרף $G \leftarrow$ קיים ניחוש למסלול המילטוני מכוון מצומת s לצומת t המובילים למצב מקבל $\leftarrow < G > \in L(N)$.
- $< G > \notin DHL \leftarrow$ לא קיים אף מסלול המילטוני מכוון בגרף $G \leftarrow$ לכל ניחוש של מסלול $s \to t$ המכונה N דוחה $\leftarrow < G > \notin L(N)$.

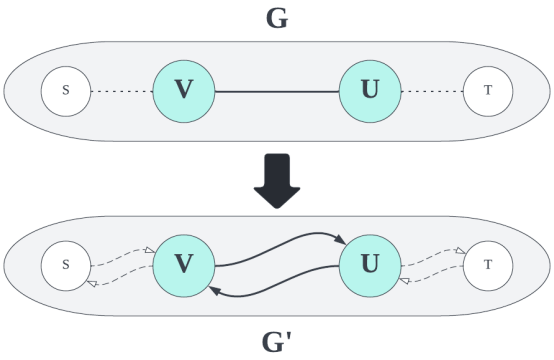
מ.ש.ל.

<u>הוכחת הרדוקציה</u>: $HL \leq_P DHL$.

<u>רעיון ההוכחה</u>: במסלול המילטוני נרצה לעבור על כל הקודקודים פעם אחת בלבד. $< G' > = f(< G >)$. בגרף G הלא מכוון קיים מסלול המילטוני, נרצה בעזרת רדוקציה להשליך גם על הגרף המכוון G' שיהיה בעל מסלול המילטוני. ניקח כל צלע בגרף הלא מכוון G, ונשים במקומה שתי צלעות מכוונות. כלומר, לכל $(u,v) \in E(G)$ ניצור 2 צלעות $(u,v),(v,u) \in E(G')$.

נרצה שאם ב-G יש מסלול המילטוני אז גם ב-G' יש, ואם ב-G אין מסלול המילטוני אז גם ב-G' אין מסלול המילטוני.

<u>ציור להמחשה</u>: מגרף G לא מכוון, לכל צלע ניצור 2 צלעות בגרף המכוון G' כך שכל אחת מ-2 הצלעות יוצאת לצומת אחרת:



חשוב לשים לב שאם לא קיים ב-G מסלול המילטוני, אז גם אם נצרף 2 צלעות, עדיין גם בגרף G' לא יהיה מסלול המילטוני וזה טוב לנו ויהיה שימושי לנכונות של תקפות הפונקציה.

<u>הוכחה</u>: נראה רדוקציה $HL \leq_P DHL$.

<u>פונקצית הרדוקציה</u>: $< G' > = f(< G >)$. כאשר G גרף לא מכוון ו-G' גרף מכוון.

<u>הפעלת הרדוקציה</u>: הגרף G' המכוון יהיה כמו G (לא מכוון) רק שעבור כל 2 קודקודים $u,v \in V(G)$ בצלע $(u,v) \in E(G)$, נעשה 2 צלעות $(u,v),(v,u) \in E(G')$.

<u>הפונקציה ניתנת לחישוב בזמן פולינומי</u>: מעבר על כל הקודקודים $O(n)$ ויצירת 2 צלעות $O(2n)$ ולכן סך הכל נקבל $O(n)$ - <u>פולינומי</u>.

<u>תקפות הפונקציה</u>: $< G > \in HL \Leftrightarrow < G' > \in DHL$.

- $< G > \in HL \leftarrow$ קיים מסלול המילטוני בגרף לא מכוון $G \leftarrow$ לכן, כאשר נשאיר את הצלע וניצור ממנו שני כיוונים, עדיין יהיה קיים מסלול המילטוני בגרף המכוון G' (זה אותו המסלול שהיה ב-G) $\leftarrow < G' > \in DHL$.
- $< G > \notin HL \leftarrow$ לא קיים מסלול המילטוני בגרף לא מכוון $G \leftarrow$ לכן, גם אם ניצור עבור כל צלע שני כיוונים עדיין לא יהיה מסלול המילטוני בגרף מכוון $G' \leftarrow < G' > \notin DHL$.

מ.ש.ל.

$$CNF - SAT \leq_P 3 - CNF - SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>רדוקציה 2</u>: $HC \leq_P HL$. תחילה צריך להוכיח כי $HL \in NP$ וגם $HC \in NP$. עבור $HL \in NP$ הוכחנו כבר בהרצאה קודמת, קראנו לשפה $HL = HAMPATH$.

<u>בניית מכונה ל-HC</u>: נוכיח כי $HC \in NP$.

<u>רעיון ההוכחה</u>: אם יהיה לנו אפשרות לנחש את הפרמוטציה שתייצג את הסידור הנכון בגרף עבור מעגל המילטון אז ניתן יהיה לפתור בזמן פולינומי.

נציג מכונה M_{HC} א"ד פולינומית המכריעה את השפה HC

<u>הוכחה</u>: המכונה M_{HC} על הקלט $< G >$ תבצע:

1. מנחשת פרמוטציה (מסלול) על הצמתים ובודקת שאכן מהווה מעגל המילטון.
2. אם כן, קבל. אחרת, דחה.

<u>סיבוכיות זמן ריצה</u>: אורך המעגל שצריך לנחש הוא ליניארי (מסלול בעץ החישוב) כלומר $O(n)$. צריך לבדוק האם הצמתים שניחשנו שונים זה מזה, ושבין כל שני צמתים עוקבים בסידור קיימת קשת - עולה לנו $O(n)$. לכן, שתי הבדיקות הללו ניתנות לביצוע בזמן פולינומי.

<u>הוכחת נכונות</u>: צריך להוכיח ש: $HC = L(M_{HC})$.

- $< G > \in HC$ ← קיים ב-G מעגל המילטון ← ניתן לתאר את המעגל ההמילטוני כסידור של צמתים ← M_{HC} תנחש סידור זה והבדיקה שלה תצליח ← קיים מסלול חישוב מקבל עבור G ← $< G > \in L(M_{HC})$.
- $< G > \notin HC$ ← לא קיים ב-G מעגל המילטון ← לא קיים אף סידור של צמתים המתאר מעגל המילטוני ← M_{HC} לא תצליח בניחוש ← לא קיים אף מסלול חישוב מקבל עבור G ← $< G > \notin L(M_{HC})$.

ומכאן $HC \in NP$ כנדרש. מ.ש.ל.

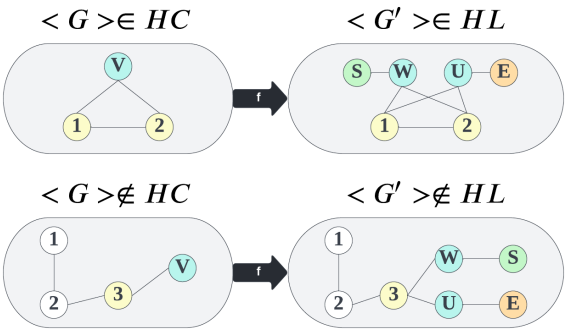
<u>הוכחת הרדוקציה</u>: $HC \leq_P HL$.

<u>רעיון ההוכחה</u>: במסלול המילטוני נרצה לעבור על כל הקודקודים פעם אחת בלבד. אנו יודעים כי כל מעגל המילטוני הוא מסלול המילטוני, אבל חשוב לשים לב שלא כל מסלול המילטוני הוא גם מעגל המילטוני!. אם ב-G יש מעגל אז הבנייה של G' קלה כי כל מעגל המילטון הוא מסלול המילטוני. אבל, בחלק של האי-שייכות, אם לא קיים מעגל המילטון ב-G אז זה לא מחייב שלא קיים מסלול המילטון ב-G' ואנחנו נרצה לבנות את G' במקרה של אי שייכות כך שלא קיים בו מסלול המילטון. $< G' > =: f(< G >)$. הגרף G הוא גרף לא מכוון בעל מעגל המילטוני, אנחנו נרצה לבנות G' שישמור על התכונות. כלומר, אנחנו נרצה להפוך מעגל למסלול: ניקח קודקוד v ו"נפצל" אותו לשני קודקודים u, w שיהיו מחוברים לכל השכנים

<u>ציור להמחשה</u>: בציור מתואר 2 התרחישים (שייכות ואי שייכות). נשים לב שאי אפשר לבנות עבור מקרה של שייך ובניה אחרת עבור מקרה שלא שייך, הבנייה שלנו צריכה לעבוד לכל מקרה.

- תרחיש 1 (שייכות - ציור עליון): אם קיים מעגל המילטון אז קיים מסלול המילטון.
- תרחיש 2 (אי שייכות - ציור תחתון): אם לא קיים מעגל המילטון אז לא קיים מסלול המילטון.

אפשר לראות בציור שבאמת עבור 2 התרחישים אנחנו עומדים בדרישות ולכן תקיפות הפונקציה שנבנה תהיה נכונה וללא באגים.



<u>הוכחה</u>: נראה רדוקציה $HC \leq_P HL$.

<u>פונקצית הרדוקציה</u>: $f(< G >) =: < G' >$.

<u>הפעלת הרדוקציה</u>: נבנה G' כך: נבחר צומת v בצורה רנדומלית ונפרק אותה ל-2 צמתים w, u, כך ש: u, v יהיו מחוברים לאותם קודקודים ש-v היה מחובר. אם לא יהיו מחוברים אז בונים לבין עצמם. כעת נוסיף שני צמתים נוספים לגרף s, e וניצור את הקשתות (e, u), (s, w).

<u>הפונקציה ניתנת לחישוב בזמן פולינומי</u>: יצירת 4 קודקודים ו-2 צלעות - פולינומי.

<u>תקפות הפונקציה</u>: $< G > \in HC \Leftrightarrow < G' > \in HL$.

- $< G > \in HC$ ← קיים מעגל המילטוני בגרף G כלומר $(v \to x \dots y \to v)$ ← בגרף G' יהיה קיים מסלול המילטוני $(s \to w \dots u \to e)$ ← $< G' > \in HL$.
- $< G > \notin HC$ ← לא קיים מעגל המילטוני בגרף G נחלק ל-2 מקרים (היה מסלול ולא היה מסלול) ←

1) היה ב-G מסלול המילטוני: אז ב-G' נבחר איזשהו v ונפרק ל-u, w ונוסף 2 קודקודים s, e שמחוברים רק ל-u, w. כדי להגיע ל-s, e נהיה חייבים להתחיל באחד מהם ולסיים בשני ולכן נשבור את הרצף של המסלול

2) לא היה ב-G מסלול המילטוני: אז גם עבור G' לא יהיה מסלול המילטוני.

$< G' > \notin HL$ ←

מ.ש.ל.

של v. נוסיף גם צלעות שיוצאות מ-w ו-u לקודקודים חדשים.

$$CNF-SAT \leq_P 3-CNF-SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>רדוקציה 3</u>: $DHC \leq_P HC$. תחילה צריך להוכיח כי $HL \in NP$ וגם $DHC \in NP$. עבור $HC \in NP$ הוכחנו ברדוקציה הקודמת, נוכיח כי $DHC \in NP$.

<u>בניית מכונה ל-DHC</u>: נוכיח כי $DHC \in NP$.

<u>רעיון ההוכחה</u>: אם יהיה לנו אפשרות לנחש את הפרמוטציה שתייצג את הסידור הנכון בגרף עבור מעגל המילטון אז ניתן יהיה לפתור בזמן פולינומי.

נציג מכונה M_{DHC} א"ד פולינומית המכריעה את השפה HC

<u>הוכחה</u>: המכונה M_{DHC} על הקלט $< G >$ תבצע:

1. מנחשת פרמוטציה (מסלול) על הצמתים ובודקת שאכן מהווה מעגל המילטון.
2. אם כן, קבל. אחרת, דחה.

<u>סיבוכיות זמן ריצה</u>: אורך המעגל שצריך לנחש הוא ליניארי (מסלול בעץ החישוב) כלומר $O(n)$. צריך לבדוק האם הצמתים שניחשנו שונים זה מזה, ושבין כל שני צמתים עוקבים בסידור קיימת קשת - עולה לנו $O(n)$. לכן, שתי הבדיקות הללו ניתנות לביצוע בזמן פולינומי.

<u>הוכחת נכונות</u>: צריך להוכיח ש: $DHC = L(M_{DHC})$.

- $< G > \in DHC$ ← קיים בגרף מכוון G מעגל המילטון ← ניתן לתאר את המעגל ההמילטוני כסידור של צמתים ← M_{DHC} תנחש סידור זה והבדיקה שלה תצליח ← קיים מסלול חישוב מקבל עבור G ← $< G > \in L(M_{DHC})$.
- $< G > \notin DHC$ ← לא בגרף מכוון G מעגל המילטון ← לא קיים אף סידור של צמתים המתאר מעגל המילטוני ← M_{DHC} לא תצליח בניחוש ← לא קיים אף מסלול חישוב מקבל עבור G ← $< G > \notin L(M_{DHC})$.

ומכאן $DHC \in NP$ כנדרש. מ.ש.ל.

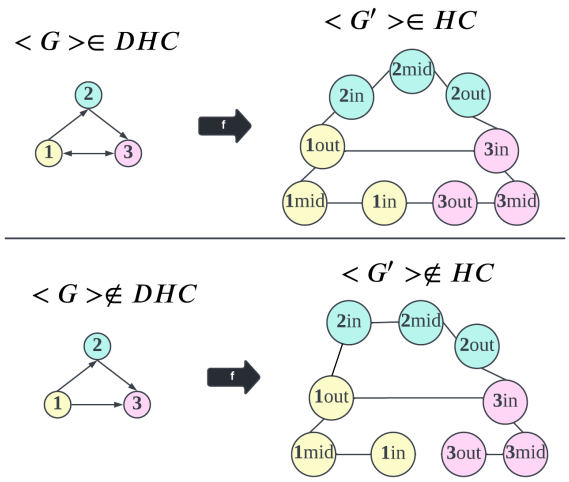
<u>הוכחת הרדוקציה</u>: $HC \leq_P DHC$.

<u>רעיון ההוכחה</u>: אם ניקח את כל הצלעות המכוונות ונהפוך אותן לצלעות לא מכוונות אז ייתכן וניצור מעגלי המילטון שלא היו קיימות וזה בעייה במקרה ונרצה להוכיח אי-שייכות!.

$< G' > = f(< G >)$. המטרה שלנו זה לעבור מגרף G מכוון לגרף G' לא מכוון ועדיין לשמר את הנכונות של הפונקציה, כלומר שאם לא קיים מעגל המילטון ב-G אז לא קיים גם ב-G' ואם כן קיים מעגל המילטון ב-G אז קיים מעגל המילטון ב-G'.

הרעיון: לכל קודקוד v בגרף המכוון G, נפצל אותו לשלושה קודקודים חדשים: $v_{in}, v_{middle}, v_{out}$.

ציור להמחשה:



<u>הוכחה</u>: נראה רדוקציה $DHC \leq_P HC$.

<u>פונקצית הרדוקציה</u>: $< G' > = f(< G >)$.

<u>הפעלת הרדוקציה</u>: נבנה G' כך: עבור כל קודקוד בגרף G נעשה 3 קודקודים $v_{in}, v_{middle}, v_{out}$. אם ב-$G$ הייתה קיימת צלע (u, v) אזי ב-G' תיהיה צלע (u_{out}, v_{in}). ואם ב-G הייתה קיימת צלע (v, u) אזי ב-G' תיהיה צלע (v_{out}, u_{in}). בנוסף נוסיף את הצלעות (u_{in}, u_{middle}), (u_{middle}, u_{out}) כדי לכפות את המעגל לעבור ב-u, כלומר ב-u_{middle}.

<u>סיבוכיות זמן ריצה</u>: שילוש כמות הקודקודים וחיבור הצלעות זה בזמן $O(n)$ - <u>פולינומי</u>.

<u>תקפות הפונקציה</u>: $< G > \in DHC \Leftrightarrow < G' > \in HC$.

- $< G > \in DHC$ ← יש ב-G מעגל המילטוני מכוון במסלול $(v_1 \to ... \to v_n \to v_1)$ ← בגרף החדש G' יש את המעגל: $(v_{1in} \to v_{1mid} \to v_{1out} ... \to v_{nin} .. \to v_{nout} \to v_{1in})$ ← $< G' > \in HC$.
- $< G' > \in HC$ ← יש ב-G' מעגל המילטוני לא מכוון ← לכל קודקוד v_{mid} מחובר רק ל-v_{in}, v_{out} וכי המעגל המילטוני אז הוא חייב לעבור $(v_{in} \to v_{mid} \to v_{out})$ או הפוך (בה"כ). ← בפרט, לא יכול להיות מצב $(w \to v_{in\backslash out} \to u)$ כאשר $u, w \neq v_{mid}$ ← לפי התיאור והבניה של הגרף, המעגל נראה כך: $(v_{1in} \to v_{1mid} \to v_{1out} ... \to v_{nin} .. \to v_{nout} \to v_{1in})$ ←

ונוסיף לו צלעות: $(v_{in}, v_{middle}), (v_{middle}, v_{out}) \in E(G')$.
ולכל צלע $(x, y) \in E(G)$ נוסיף את $(x_{out}, y_{in}) \in E(G')$.

בגרף המקורי G יש מעגל $\left(v_1 \to \ldots \to v_n \to v_1\right) \leftarrow$
$< G > \in DHC$.

מ.ש.ל.

$$CNF - SAT \leq_P 3 - CNF - SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>רדוקציה 5</u>: $3-CNF-SAT \leq_P VC$. תחילה צריך להוכיח כי $VC \in NP$ וגם $3-CNF-SAT \in NP$. עבור $VC \in NP$ הוכחנו בהרצאה, נוכיח כי $3-SAT \in NP$.

<u>בניית מכונה ל-3SAT</u>: נוכיח כי $3-SAT \in NP$.

<u>רעיון ההוכחה</u>: נבנה N_3 מ"ט א"ד פולינומית שמכריעה את השפה $3-CNF-SAT$. אין לנו צורך לבצע בדיקה שאכן הפסוק ϕ מכיל 3 ליטרלים בכל פסוקית, אנחנו מניחים שזה תקין. הבדיקה היחידה שיש לנו לעשות הוא עבור ההשמה שננחש, נרצה לבדוק שההשמה הזאת אכן מספקת את ϕ.

<u>הוכחה</u>: המכונה N_3 על הקלט $< \phi >$ תבצע:

1. נחש השמה π.
2. נבדוק שאכן $\phi(\pi) = true$.
3. אם כן, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>סיבוכיות זמן ריצה</u>: נסמן ב-n את מספר המשתנים ב-ϕ.

1. שורה 1: כגודל הניחוש $O(|\pi|) = O(n)$.
2. שורה 2: הצבה לבדיקת הסיפוק $O(|\phi|)$.

סה"כ $O(|\phi|)$ - <u>פולינומי</u>.

<u>הוכחת נכונות</u>: צריך להוכיח ש: $L(N_3) = 3SAT$.

- $< \phi > \in 3SAT$ ← קיימת השמה π שמספקת את הפסוק ϕ ← קיים ניחוש של π שיוביל למצב מקבל ← $< \phi > \in L(N_3)$.
- $< \phi > \notin 3SAT$ ← לכל השמה π, היא לא תספק את ϕ ← לכל ניחוש נדחה ← $< \phi > \notin L(N_3)$.

ומכאן $3-CNF-SAT \in NP$ כנדרש.

מ.ש.ל.

<u>הוכחת הרדוקציה</u>: $3-CNF-SAT \leq_P VC$.

<u>רעיון ההוכחה</u>: הרדוקציה ממירה פסוק $3cnf$ שנסמנה ב-ϕ לגרף G ומספר k, כך ש ϕ ספיק כאשר G מכיל כיסוי קודקודים של k קודקודים. ההמרה נעשית ללא הידיעה אם ϕ ספיק או לא. בפועל, G מסמלץ את ϕ. הגרף מכיל כלים שמחקים (חיקוי) את המשתנים והפסוקיות של ϕ. אנחנו ניצור 2 קבוצות בגרף שנבנה: קבוצת קודקודי המשתנים וקבוצת קודקודי הפסוקיות.

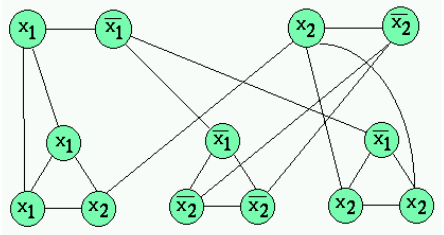
1. <u>קבוצת קודקודי הפסוקיות</u>: עבור כל פסוקית $(x \vee y \vee z)$, נגדיר שלושה קודקודים המחוברים ביניהם.
2. <u>קבוצת קודקודי המשתנים</u>: עבור כל משתנה, נגדיר שני קודקודים מחוברים (צלע), עבורו ועבור ההופכי שלו.

לאחר מכן, נחבר בין 2 הקבוצות שהצגנו, כלומר נחבר צלעות בין ליטרלים זהים בשתי הקבוצות - כל משתנה בקבוצת המשתנים נחבר בצלע עם אותו המשתנה שנמצא בקבוצת הפסוקיות. אנחנו נבחר את k להיות $k = m + 2l$ כאשר m זה מספר המשתנים ו-l זה מספר הפסוקיות.

<u>ציור להמחשה</u>: נסתכל לדוגמה על הפסוק הבא:

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

הרדוקציה תייצר $< G, k >$ מ-ϕ, כאשר $k = m + 2l = 8$



<u>הוכחה</u>: נראה רדוקציה $3-CNF-SAT \leq_P VC$.

<u>פונקצית הרדוקציה</u>: $f(< \phi >) = < G, k >$.

<u>הפעלת הרדוקציה</u>: לכל משתנה x ב-ϕ, ניצור צלע המחבר שני קודקודים. אנחנו נסמן את שני הקודקודים האלה ב-x ו-$\bar{x}$. כאשר $x = TRUE$, וגם $\bar{x} = FALSE$. כל פסוקית היא שלשה של קודקודים המסומנים בהתאם לשלושת הליטרלים בפסוקית הזאת. שלושת הקודקודים האלה מחוברים ביניהם וגם מחוברים לקודקודי המשתנים שלהם בהתאם לסימון. נסמן ב-m את המשתנים ו-l את הפסוקיות. מספר הקודקודים בגרף G הוא $2m + 3l$ וגם $k = m + 2l$.

<u>סיבוכיות זמן ריצה</u>: לכל משתנה ניצור 2 קודקודים עם צלע מחבר ביניהם, לכל פסוקית ניצור 3 קודקודים עם 3 צלעות ביניהם. בנוסף, לכל ליטרל זהה ב-2 הקבוצות נחבר בצלע. סה"כ - זמן פולינומי.

<u>תקפות הפונקציה</u>: $< \phi > \in 3SAT \Leftrightarrow < G, k > \in VC$.

- $< \phi > \in 3SAT$ ← יש השמה מספקת ל-ϕ ← נבנה כיסוי בגודל $k = m + 2l$:
  - מקודקודי המשתנים, ניקח מכל זוג את הליטרל עם ה-T.
  - מקודקודי הפסוקיות, נבחר מכל משולש שני קודקודים, כך שהשלישי קיבל ערך T (חייב להיות כזה כי ההשמה עבור ϕ היא מספקת, לכן חייב להיות לפחות T אחד).

  ← קיבלנו כיסוי, כי הצלעות מקבוצת המשתנים מכוסות, הצלעות בתוך המשולשים מכוסות והצלעות בין הקבוצות מכוסות: עבור משתנים שקיבלו T צד המשתנים וגם עבור משתנים שקיבלו F בצד המשולשים. ← $< G, k > \in VC$.
- $< G, k > \in VC$ ← קיים כיסוי קודקודים בגודל $k = m + 2l$ בגרף G ← כדי לכסות את כל הצלעות במשולשים (לבנות פסוקיות עם 3 ליטרלים), הכיסוי חייב לקחת לפחות 2 קודקודים מכל משולש ← כדי לכסות את הצלעות במשתנים, הכיסוי חייב לקחת לפחות קודקוד אחד עבור כל משתנה. ← נבנה השמה: לכל משתנה, נחבר שיהיה T אם הקודקוד שלו בכיסוי, ו-F אם השלילה בכיסוי. כיוון שהראנו שרק אחד נבחר מכל משתנה, זו השמה חוקית. ← ההשמה שבנינו מספקת: בכל משולש, הקודקוד שלא נבחר לכיסוי חייב להתחבר לקודקוד משתנה שכן נבחר (אחרת הצלע לא מכוסה), ולכן קיבל ערך T ומכאן ← $< \phi > \in 3SAT$.

$$CNF-SAT \leq_P 3-CNF-SAT \leq_P VC \leq_P DHC \leq_P HC \leq_P HL \leq_P DHL$$

<u>רדוקציה 6</u>: $3-CNF-SAT \leq_P CNF-SAT$.

לנוחות, נסמן במקום ב- $3SAT \leq_P SAT$. תחילה צריך להוכיח כי $3SAT \in NP$ וגם $SAT \in NP$. הוכחנו בהרצאה $SAT \in NP$ וגם הוכחנו ברדוקציה הקודמת כי $3SAT \in NP$.

<u>הוכחת הרדוקציה</u>: $3SAT \leq_P SAT$.

<u>רעיון ההוכחה</u>: אנחנו צריכים להמיר כל פסוקית מהפסוק המקורי $\phi \in SAT$ לפסוקיות בגודל 3 בפסוק $\phi' \in 3SAT$. אנחנו נראה רדוקציה פולינומית מ-SAT ל-$3SAT$, ואז נוכל להוכיח ש: $3SAT$ היא NP-שלמה. מספיק להראות איך לתרגם פסוקית אחת לפסוק $3CNF$. עבור פסוק כללי, אפשר להפעיל את התהליך על כל פסוקית לחוד. ההמרה תתבצע בצורה הבאה:

1. לכל פסוקית שיש בה 1 או 2 ליטרלים, אנחנו נבחר אחד מהם ונשכפל אותו בתוך הפסוקית עד שנגיע ל-3 ליטרלים.
2. לכל פסוקית שיש בה יותר מ-3 ליטרלים, אנחנו נפצל אותה לכמה פסוקיות בגודל 3, במידה ונשאר לנו שארית להשלים אז נבחר במשתנה במשתני עזר y_i ונשכפל אותם עד שנגיע ל-3 ליטרלים בכל הפסוקיות.

<u>דוגמה להמחשה</u>: נסמן ליטרל ב-l ופסוקית ב-p.

$$(l_1) \Rightarrow (l_1 \vee l_1 \vee l_1)$$
$$(l_1 \vee l_2) \Rightarrow (l_1 \vee l_2 \vee l_2)$$
$$(l_1 \vee l_2 \vee l_3) \Rightarrow (l_1 \vee l_2 \vee l_3)$$

עבור פסוקית p כך ש: $n \geq 4$, הפתרון מורכב יותר ודורש שימוש במשתני עזר שנסמן ב-$y_1, y_2, \ldots, y_{n-3}$.

באופן כללי, אם הפסוקית p מכילה n ליטרלים:

$$p = (l_1 \vee l_2 \vee \ldots \vee l_n)$$

אז אנחנו יכולים להחליף אותה עם $n-2$ פסוקיות חדשות וגם $n-3$ משתנים חדשים בנוסף לקיימים.

לדוגמה, נתבונן בפסוקית הבאה:

$$(l_1 \vee l_2 \vee l_3 \vee l_4) \Rightarrow (l_1 \vee l_2 \vee y_1) \wedge (\bar{y_1} \vee l_3 \vee l_4)$$

יש לה $n = 4$ ליטרלים, איתה ניצור $n-2 = 2$ פסוקיות חדשות ונוספו לה $n-3 = 1$ משתנים חדשים (y_1).

ככה אנחנו נפעל לכל פסוקית שנפגוש בתהליך ההמרה כך שנבדיל בין פסוקיות עם לכל היותר 3 ליטרלים ופסוקיות עם לפחות 4 ליטרלים.

<u>הוכחה</u>: נראה רדוקציה $SAT \leq_P 3SAT$.

<u>פונקצית הרדוקציה</u>: $f(<\phi>) = <\phi'>$.

<u>הפעלת הרדוקציה</u>: נבצע המרה בין פסוק ϕ רגיל לפסוק ϕ' מצורת $3-CNF$. נסמן ב-n_i את מספר הליטרלים הנמצאים בפסוקית p_i ונבצע המרה בצורה הבאה:

1. לכל פסוקית בגודל $n \leq 2$: נבחר ליטרל מהפסוקית ונשכפל אותו עד שנגיע ל-3 ליטרלים בפסוקית.
2. לכל פסוקית בגודל $n = 3$: נשאיר אותו כמו שהוא.
3. לכל פסוקית בגודל $n \geq 4$: נפצל את הפסוקית ל-$n-2$ פסוקיות חדשות המכילות את הליטרלים הקיימים, יחד עם $n-3$ משתני עזר y חדשים ונוספים הממלאים את הפסוקיות החדשות לצורה של 3-ליטרלים (במידה וחסר).

<u>סיבוכיות זמן ריצה</u>: בזמן הגרוע, עבור כל פסוקית בגודל n, נוסיף $n-3$ ליטרלים ו-$n-2$ פסוקיות חדשות ולכן זה מתבצע בזמן <u>פולינומי</u>.

<u>תקפות הפונקציה</u>: $<\phi> \in SAT \Leftrightarrow <\phi'> \in 3SAT$.

- $<\phi> \in SAT \rightarrow$ קיימת ל-ϕ השמה מספקת $\pi \rightarrow$ נבנה השמה מספקת לכל פסוקית חדשה $\rightarrow$ עבור המשתנים המקוריים, ההשמה תיתן להם את אותו הערך $\pi'(l) = \pi(l)$. $\rightarrow$ ההשמה מספקת ולכן יש לפחות ליטרל אחד l_i שיקבל ערך $T \rightarrow$ עבור משתנים חדשים נבצע לכל $i \leq j-2$ את $\pi'(l_j) = T$ וגם נבצע לכל $i > j-2$ את $\pi'(l_j) = F \rightarrow$ כל הפסוקיות יסתפקו ע"י המשתנים החדשים, מלבד הפסוקית של l_i, שתסתפק כי הוא ערך $T \rightarrow$ יש השמה מספקת ל-$\phi' \rightarrow <\phi'> \in 3SAT$.
- $<\phi'> \in 3SAT \rightarrow$ קיימת השמה מספקת ל-$\phi' \rightarrow$ נניח בשלילה ש $\phi'(l_i) = \phi(l_i)$ לא השמה מספקת ל-$\phi \rightarrow$ אז כל הליטרלים קיבלו ערך $F \rightarrow T = \phi(y_i)$ כי ההשמה צריכה לספק את $\phi' \rightarrow$ הפסוקית האחרונה לא מסתפקת $\rightarrow$ **סתירה**. $\rightarrow$ לכן, $\phi'(l_i) = \phi(l_i)$ זו השמה מספקת ל-$\phi \rightarrow <\phi> \in SAT$.

מ.ש.ל.

תרגיל 3 - השמה מאוזנת - תרגול 6

נאמר על השמה שהיא **מאוזנת** אם מספר המשתנים שקיבלו T שווה למספר המשתנים שקיבלו F.

נתונה השפה: $BALANCED-SAT=\{\phi \mid \exists\ balanced\ satisfying\ assignments\ for\ \phi\}$.

הוכיחו: $BALANCED-SAT\in NPC$.

<u>פתרון</u>: נוכיח כי $BALANCED-SAT\in NP$ ולאחר מכן נוכיח כי $BALANCED-SAT\in NPH$.

<u>פתרון - שלב א'</u>: נוכיח כי $BALANCED-SAT\in NP$.

נבנה N מ"ט א"ד פולינומית המכריעה את השפה $BALANCED-SAT$.

N על קלט $<\phi>$ תבצע:

1. אם מספר המשתנים ב-ϕ הוא אי-זוגי - <u>דחה</u>.
2. נחש השמה π.
3. בדוק שההשמה מאוזנת - אם לא, <u>דחה</u>.
4. בדוק שמתקיים $\phi(\pi)=true$.
5. אם כן, <u>קבל</u>. אחרת, <u>דחה</u>.

<u>סיבוכיות זמן הריצה</u>: נסמן את מספר המשתנים של הפסוק ב-n.

1. שלב 1 - בדיקת מספר המשתנים הוא $O(n)$.
2. שלב 2 - כגודל הניחוש: $O(n)$.
3. שלב 3 - מעבר על כל המשתנים כדי לבדוק האם הם מאוזנים: $O(n)$.
4. שלב 4 - בדיקת הסיפוק ע"י הצבה במעבר על הפסוק: $O(|\phi|)$.
5. שלב 5 - בדיקה בזמן קבוע $O(1)$.

סך הכל קיבלנו זמן ריצה של כ- $O(|\phi|)$ ולכן <u>פולינומי</u>.

<u>הוכחת נכונות הבניה</u>: נוכיח כי $L(N)=BALANCED-SAT$.

- $<\phi>\in BALANCED-SAT$ ← קיימת השמה מספקת ומאוזנת עבור הפסוק ϕ ← קיים ניחוש π שיוביל למצב מקבל ← $<\phi>\in L(N)$.
- $<\phi>\notin BALANCED-SAT$ ← לכל השמה π קיימים 3 מקרים אפשריים:
  - מספר המשתנים הוא אי זוגי.
  - מספר המשתנים שקיבלו T שונה למספר המשתנים שקיבלו F.
  - ההשמה π לא מספקת את ϕ.

← לכל ניחוש נקבל לפחות אחד מהמקרים שהצגנו ולכן N תדחה ← $<\phi>\notin L(N)$.

<u>פתרון - שלב ב'</u>: נוכיח כי $BALANCED-SAT\in NPH$ בעזרת רדוקציה מ-SAT.

<u>תיאור הרדוקציה</u>: אנו יודעים כי $SAT\in NPH$, לכן נראה רדוקציה $SAT\leq_P BALANCED-SAT$.

- פונקצית הרדוקציה: $f(<\phi>)=<\phi'>$. כאשר ϕ פסוק רגיל ו-ϕ' פסוק עם מספר משתנים זוגי.
- ביצוע: נבנה את הפסוק ϕ' כך שהוא מכיל את הפסוק ϕ ויחד איתו מחוברים n פסוקיות (ב-AND) טאוטולוגיות (תמיד נכונות). כלומר $\phi'=\phi\wedge(y_1\vee\bar{y_1})\wedge\ \dots\ \wedge(y_n\vee\bar{y_n})$.

הפונקציה פולינומית: אנחנו מוסיפים מספר קבוע של פסוקיות טאולוגיות - פולינומי.

הפונקציה תקפה: נראה שמתקיים $< \phi >\in SAT \Leftrightarrow < \phi' >\in BALANCED - SAT$.

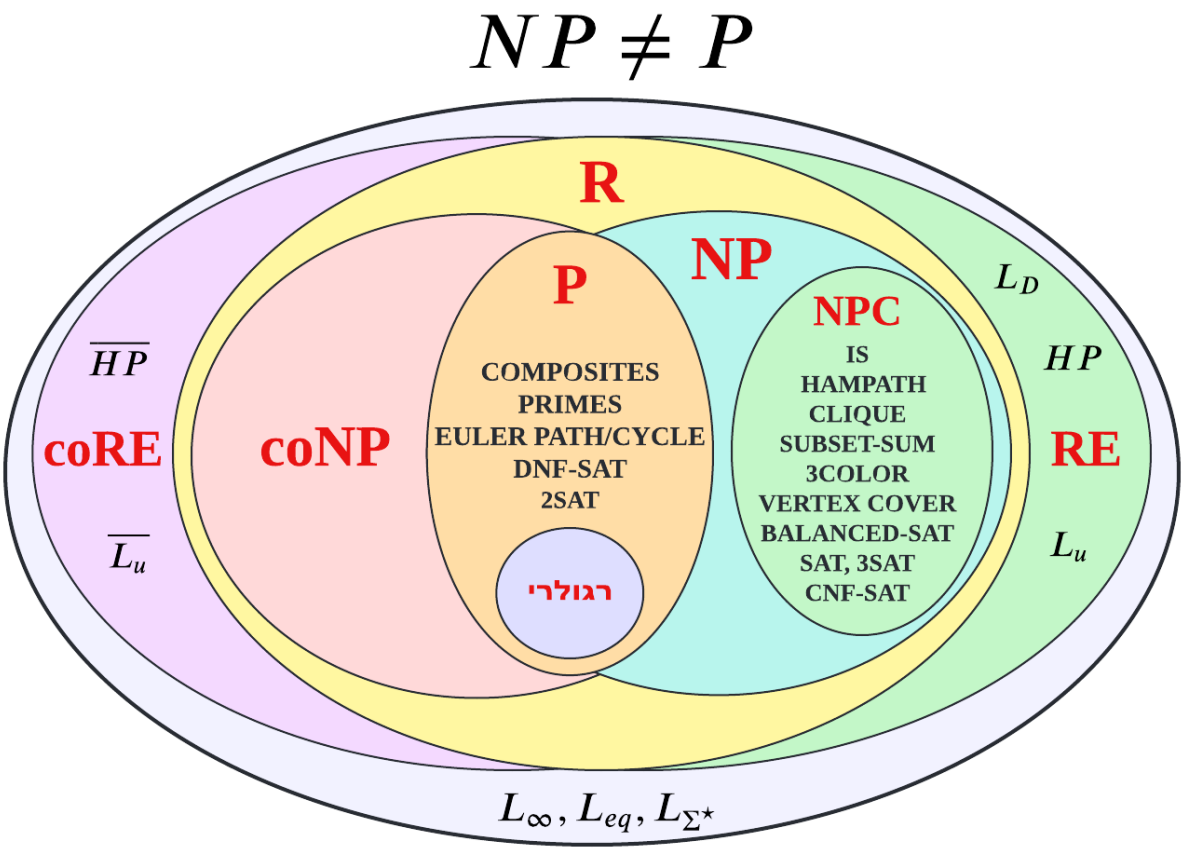
- $< \phi >\in SAT$ ← קיימת השמה מספקת ל-ϕ ← לכל הצבה של ערכים ל-y-ים, אנחנו נשאר עם השמה מספקת (כי זה טאוטולוגי) ← נבחר לכל y_i את הערך $\bar{x}_i$ ← נקבל השמה מספקת ומאוזנת ← $< \phi' >\in BALANCED - SAT$.
- $< \phi' >\in BALANCED - SAT$ ← קיימת השמה מספקת ← $< \phi >\in SAT$.

לכן $BALANCED - SAT \in NPC$ כנדרש.

■

תמונת מצב

לחלק מהשפות בציור עוד לא הוכחנו שייכות אבל נוכיח בהמשך... נשים לב ש: $L_\infty, L_{eq}, L_{\Sigma^\star} \in \overline{RE \cup coRE}$.



דוגמאות לשפות NP-שלמות

מבוא

בפרק הקודם הכרנו שתי שפות NP-שלמות:

$$SAT = \{< \phi > \mid \phi \; is\; a\; satisfiable\; Boolean\; formula\}$$
$$CNF - SAT = \{< \phi > \mid \phi \; is\; a\; satisfiable\; CNF\; Boolean\; formula\}$$

כעת נלמד כמה שפות נוספות שהם גם בעיות NP-שלמות בכל מיני תחומים (גרפים, לוגיקה, קבוצות וכו'..)

<u>איך נוכיח על שפות נוספות</u>?: ברגע שידועה לנו שפה NP-שלמה אחת, אפשר להוכיח על שפות נוספות שגם הן NP-שלמות.

משפט

אם $L_1 \in NPC$ וגם L_2 מקיימת את התכונות הבאות:

1. $L_2 \in NP$.
2. $L_1 \leq_P L_2$, כלומר L_1 ניתנת לרדוקציה עם L_2.

אז גם $L_2 \in NPC$.

<u>תרגיל</u>: הוכיחו את המשפט המתואר מעל, זכרו כי היחס $\leq_P$ הוא טרנזיטיבי.

<u>הוכחה</u>: נתון לנו כי $L_2 \in NP$, לכן אנחנו חייבים להראות לכל שפה $L' \in NP$ ניתנת לרדוקציה פולינומית עם L_2. מאחר ש: $L_1 \in NPC$, כל שפה $L' \in NP$ ניתנת לרדוקציה פולינומית $L'\leq_P L_1$, וגם נתון לנו כי $L_1 \leq_P L_2$.
לכן, מאחר והיחס $\leq_P$ הוא טרנזיטיבי, אז כל $L' \in NP$ ניתנת לרדוקציה פולינומית עם L_1 וגם L_1 ניתנת לרדוקציה פולינומית עם L_2. לכן, כל שפה $L' \in NP$ ניתנת לרדוקציה פולינומית עם L_2 ומכאן $L_2 \in NPC$.

∎

השפה 3SAT

<u>הגדרה</u>: פסוק בתחשיב הפסוקים הוא $3CNF$ אם:

1. הפסוק CNF.
2. בכל פסוקית יש שלושה ליטרלים.

השפה: $3SAT = \{< \phi > \mid \phi \; is\; a\; satisfiable\; 3CNF\; Boolean\; formula\}$.
כלומר, זוהי שפת הפסוקים הספיקים ב-$3CNF$.

<u>דוגמה</u>: $(x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee w)$.

<u>משפט</u>: $3SAT \in NPC$.

<u>הוכחה</u>: נראה $3SAT \leq_p CNF - SAT$.

<u>פונקצית הרדוקציה</u>: $f(\phi(x_1,...,x_n)) = \phi'(x_1,...,x_n)$. כל פסוקית ב-$\phi$ נמיר לפסוקית אחת או יותר בעלות שלושה ליטרלים ב-ϕ.

<u>תיאור הרדוקציה</u>: נחלק את הפסוקיות ב-ϕ לשלושה סוגים ונטפל בכל סוג בנפרד:

1. פסוקיות עם פחות מ-3 ליטרים - פסוקיות כאלה פשוט נשכפל את אחד הליטרלים עד שנגיע ל-3 ליטרלים.
2. פסוקיות עם 3 ליטרלים בדיוק - לא נשנה אותם.
3. פסוקיות עם לפחות 4 ליטרלים - מצריך עבודה עדינה יותר: עבור כל פסוקית עם איזשהו $k \geq 4$ ליטרים, נוסיף עוד $k-3$ משתנים חדשים המסומנים ב-y, ונפצל את הפסוקית ל-$k-4$ פסוקיות בגודל 3 באופן הבא:
   a. הפסוקית המקורית: $(l_1 \vee l_2 \vee l_3 \vee l_4 \vee ..... \vee l_{k-2} \vee l_{k-3} \vee l_k)$
   b. הפסוקיות החדשה:

$$(l_1 \vee l_2 \vee y_1) \wedge (\bar{y_1} \vee l_3 \vee y_2) \wedge (\bar{y_2} \vee l_4 \vee z_3) \wedge ..... \wedge (\bar{y}_{k-3} \vee l_{k-1} \vee l_k)$$

<u>סיבוכיות זמן ריצה</u>: בה"כ נניח של-ϕ יש n משתנים ו-m פסוקיות.

1. <u>המרת סוג 1</u>: כגודל הפסוקיות: $O(m)$.
2. <u>המרת סוג 2</u>: לא נשנה דבר - זמן קבוע: $O(1)$.
3. <u>המרת סוג 3</u>: יש לכל היותר m פסוקיות המצריכות טיפול כזה ובכל פסוקית יש לכל היותר n ליטרלים ולכן $O(n \cdot m)$.

סה"כ קיבלנו זמן $O(m + n \cdot m) = O(n \cdot m)$ - <u>פולינומי</u>.

<u>הוכחת נכונות</u>:נוכיח שהפונקציה שיצרנו היא אכן רדוקציה, כלומר: $< \phi > \in SAT \Leftrightarrow < \phi' > \in 3SAT$.

מכיוון שטיפלנו בכל פסוקית בנפרד, נרצה להוכיח שכל פסוקית ב-ϕ אם"ם הפסוקית או הפסוקיות המקבילות לה ב-ϕ' ספיקות. מקרים 1+2 טריוויאלים ולכן נפרט רק על מקרה 3.

נראה כי ϕ ספיקה $\Leftrightarrow$ ϕ' ספיקה:

<u>מצד אחד - אם הפסוק ספיק</u>: אז הפסוקית הגדולה ספיקה, אז יש בה לפחות ליטרל אחד חיובי.

נפצל שוב לשלושה מקרים משלימים:

1. אם הליטרל החיובי נמצא בפסוקית הראשונה שנוצרה, אז נוכל להגדיר שלכל $j \in [1, k-3]$ יתקיים $y_j = F$ ולכן כל הפסוקיות שנוצרו יסתפקו.

$$Z_i = (l_{i,1} + l_{i,2} + y_1) \cdot (\overline{y_1} + l_{i,3} + y_2) \cdots (\overline{y_{j-3}} + l_{i,j-1} + y_{j-2}) \cdot (\overline{y_{j-2}} + l_{i,j} + y_{j-1}) \cdot (\overline{y_{j-1}} + l_{i,j+1} + y_j) \cdots (\overline{y_{k-4}} + l_{i,k-2} + y_{k-3}) \cdot (\overline{y_{k-3}} + l_{i,k-1} + l_k)$$

2. אם הליטרל החיובי נמצא בפסוקית האחרונה שנוצרה, אז נוכל להגדיר שלכל $j \in [k-3]$ יתקיים $y_j = T$ ולכן כל הפסוקיות שנוצרו יסתפקו.

$$Z_i = (l_{i,1} + l_{i,2} + y_1) \cdot (\overline{y_1} + l_{i,3} + y_2) \cdots (\overline{y_{j-3}} + l_{i,j-1} + y_{j-2}) \cdot (\overline{y_{j-2}} + l_{i,j} + y_{j-1}) \cdot (\overline{y_{j-1}} + l_{i,j+1} + y_j) \cdots (\overline{y_{k-4}} + l_{i,k-2} + y_{k-3}) \cdot (\overline{y_{k-3}} + l_{i,k-1} + l_k)$$

3. אם הליטרל החיובי נמצא באמצע, כלומר בפסוקית ה-i, אז נוכל להגדיר שלכל $j \in [k-2]$ יתקיים $y_j = T$ ושלכל $i-3 \leq j \leq k-3$ יתקיים $y_j = F$ ולכן כל הפסוקיות שנוצרו יסתפקו.

<u>מצד שני - אם הפסוק לא ספיק</u>: נרצה להוכיח שגם צירוף הפסוקיות הבאות לא יסתפק.
מאחר והפסוקיות המקוריות לא הסתפקה, כל הליטרלים בה שליליים.
נניח בשלילה שהוא כן ספיק, אם כך, בהכרח $y_1 = T$ ולכן בהכרח $y_2 = T$ ונמשיך ככה עד $y_{k-3} = T$ ונקבל שהפסוקית אחרונה לא מסתפקת! - סתירה.

∎

<u>הערה</u>: ניתן להגדיר את $3CNF$ כך שאין פסוקיות בהן משתנה מופיע יותר מפעם אחת. במצב כזה, עבור פסוקית בעלת פחות מ-3 משתנים, נצטרך לבצע את הרדוקציה אחרת. עבור פסוקית עם שני משתנים $(l_1 \vee l_2)$ ניצור שתי פסוקיות חדשות: $(l_1 \vee l_2 \vee \bar{x}) \wedge (l_1 \vee l_2 \vee x)$.

<u>תרגיל 1</u>: מה יקרה עבור פסוקית עם משתנה יחיד?
<u>פתרון 1</u>: אם קיים פסוקית C_i עם משתנה (ליטרל) אחד $C_i = (l_i)$ אז ניצור עבורו 2 משתנים חדשים $y_{i,1}$, $y_{i,2}$ נחליף את C_i בצירוף פסוקיות Z_i כאשר $Z_i = (l_i \vee y_{i,1} \vee y_{i,2}) \wedge (l_i \vee \overline{y_{i,1}} \vee y_{i,2}) \wedge (l_i \vee y_{i,1} \vee \overline{y_{i,2}})$.

<u>תרגיל 2</u>: מה הסיבוכיות של ההמרות הללו?
<u>פתרון 2</u>: **להשלים**.

<u>תרגיל 3</u>: המירו את הפסוק הבא לצורת $3CNF$:

$$(x_1 \vee x_2 \vee \bar{x}_3 \vee x_4 \vee \bar{x}_5 \vee x_6) \wedge (x_3) \wedge (x_1 \vee \bar{x}_4) \wedge (\bar{x}_3 \vee x_4 \vee \bar{x}_6)$$

<u>תרגיל 3</u>: **להשלים**.

השפה 2SAT

הגדרה: פסוק בתחשיב הפסוקים הוא $2CNF$ אם:

1. הפסוק CNF.
2. בכל פסוקית יש שני ליטרלים.

השפה: $2SAT = \{< \phi > \mid \phi \; is \; a \; satisfiable \; 2CNF \; Boolean \; formula\}$.

כלומר, זוהי שפת הפסוקים הספיקים ב-$2CNF$.

דוגמה: $(x \vee \neg y) \wedge (\neg x \vee y)$.

שאלה: האם $2SAT \in NPC$?

תשובה: אי אפשר!, נוכיח כי $2SAT \in P$!

משפט: $2SAT \in P$

מבוא להוכחה: במקרה של פסוקית שיש בה רק שני ליטרלים, אם ידוע לנו על אחד מהם שהוא F אז השני חייב להיות T כדי שהפסוק יסתפק. מה שאומר שניתן לבנות גרף מכוון שירכז את כל התנאים של כל הפסוקיות ולהשתמש בגרף הנ"ל כדי למצוא השמה מספקת. כך שאם השמה זו לא קיימת, נוכל לזהות זאת ולהחזיר תשובה שלילית.

הוכחה: נוכיח ע"י בניית גרף שייצג שקילות לוגית לפסוק. הבנייה תעשה בזמן פולינומי, הפעולות על הגרף ייעשו בזמן פולינומי, סה"כ נקבל אלגוריתם פולינומי הבודק האם פסוק $2CNF$ ספיק.

הפסוק שנעסוק בו הוא $\phi(x_1,...,x_n) = C_1 \wedge C_2 \wedge \; ... \; \wedge C_m$ בעל m פסוקיות.

תזכורת 1: פסוקית מהצורה $(l_i \vee l_j)$ שקולה לוגית לפסוק $\bar{l}_i \to l_j$ ולפסוק $\bar{l}_j \to l_i$.

תזכורת 2: $\neg\neg x = x$.

המשך: נייצג כל פסוקית $(l_i \vee l_j)$ בגרף ע"י השקילות הלוגית $(\bar{l}_j \to l_i) \wedge (\bar{l}_i \to l_j)$.

נרצה לבנות את הגרף $G = (V, E)$ באופן הבא:

- קבוצת הקודקודים: $V = \{x_i, \bar{x}_i : i \in [n]\}$.
- קבוצת הצלעות: $E = \{\bar{x}_i \to x_j \; , \; \bar{x}_j \to x_i \mid (x_i \vee x_j) \in \phi\}$.
- בגרף יהיו $2n$ קודקודים וגם $2m$ צלעות ולכן קידוד הגרף יצריך זמן פולינומי בגודל הפסוק.

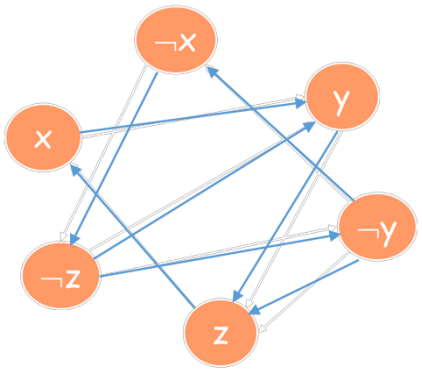
אבחנה 1: נשים לב שהגרף סימטרי, כלומר אם יש קשת $a \to b$ אז יש גם קשת $\bar{b} \to \bar{a}$. (זה נובע ישירות מכך שאם יש קשת $a \to b$ אז מופיעה הפסוקית $(\bar{a} \vee b)$ ולכן בגרף ישנה הצלע $\bar{b} \to \bar{a}$).

<u>הסבר בעברית (תכל'ס) לבניית הגרף שהצגנו</u>: ניצור גרף $G = (V, E)$ עם $2n$ קודקודים. אינטואיטיבית, לכל משתנה ניצור 2 קודקודים, קודקוד ראשון לעצמו וקודקוד שני לשלילה שלו. לכל פסוקית $(a \vee b)$ כאשר a, b הם ליטרלים, ניצור צלע מכוונת $(\bar{a}, b)$ וגם $(\bar{b}, a)$. הצלעות האלה אומרות בעצם שאם a הוא F, אז b חייב להיות T (להפך). מכאן, קיים צלע מכוונת $(a, b) \in E(G)$ אם"ם קיים פסוקית $(\bar{a}, b) \in \phi$.

<u>דוגמה</u>: נסתכל על הפסוק הבא:

$\phi = (\bar{x} \vee y) \wedge (\bar{y} \vee z) \wedge (x \vee \bar{z}) \wedge (z \vee y)$.

אפשר לראות למשל עבור הפסוקית הראשונה $(\bar{x} \vee y)$ אז אנחנו צריכים ליצור צלע $(\bar{x}, y) = (x, y)$ וגם צלע $(\bar{y}, \bar{x})$, אפשר לראות בציור כי באמת קיימות הצלעות המכוונות האלה.



<u>המשך</u>: אם נשתמש באבחנה זו מספר פעמים, נשים לב שמסלול:

$v_1 \to v_2 \to \ldots \to v_{k-1} \to v_k$ גורר את הצלעות:

$\bar{v}_k \to \bar{v}_{k-1} \to \ldots \to \bar{v}_2 \to \bar{v}_1$, $\bar{v}_3 \to \bar{v}_2$, ... , $\bar{v}_2 \to \bar{v}_1$ שיגררו את קיום המסלול: $\bar{v}_k \to \bar{v}_{k-1} \to \ldots \to \bar{v}_2 \to \bar{v}_1$.

<u>אבחנה 2</u>: לכל צלע $u \to v$ מתקיים: אם הקודקוד u קיבל T אז גם הקודקוד v צריך לקבל T.

למה? כי $(\bar{u} \vee v) \equiv u \to v$.

<u>נסח קריטריון כללי</u>: אין השמה מספקת לפסוק **אם"ם** קיים משתנה x כך שקיים מעגל מכוון:

1. קיים מסלול מ-x ל-$\bar{x}$ בגרף.
2. קיים מסלול מ-$\bar{x}$ ל-x בגרף.

<u>הוכחת הקריטריון</u>:

- <u>כיוון א'</u> - אם לא קיים בגרף משתנה x כך ש: x וגם $\bar{x}$ נמצאים שניהם על מעגל מכוון אחד, אזי קיימת השמה מספקת לפסוק. נניח שלא קיים קודקוד x כך שיש מסלול מ-x לעבר $\bar{x}$, וגם יש מסלול מ-$\bar{x}$ לעבר x. נחפש קודקוד (ליטרל) שנסמן ב-$\bar{u}$ אשר יש ממנו מסלול אל שלילתו. כלומר $u \to \ldots \to \bar{u}$, אם לא קיים מסלול כזה, נבחר קודקוד כלשהו.
  - ניתן לליטרל $u = T$ ולליטרל $\bar{u} = F$.
  - לכל הקודקודים שיש מסלול אליהם מ-u, ניתן ערך T ו-F לשלילתם.

  יש שתי בעיות שעלולות לצוץ בשיטה זו:

  1. יש מסלול מ-u לקודקוד כלשהו w וגם מסלול מ-u ל-$\bar{w}$: בגלל הסימטריה של הגרף, מכיוון שיש מסלול מ-u אל w אז גם יש מסלול מ-$\bar{w}$ אל $\bar{u}$, כלומר יש מעגל המכיל את u וגם את $\bar{u}$, בסתירה להנחה.

2. יש קודקוד שנרצה לתת לו ערך T אבל הוא כבר קיבל ערך F: זה יקרה אם $\bar{w}$ קיבל T. כלומר יש מסלול מ-u אל $\bar{w}$. שוב, בגלל הסימטריה שלה הגרף, נובע שיש מסלול מ-w אל $\bar{u}$, בסתירה להנחה.

- כיוון ב' - אם קיים בגרף משתנה x כך ש: x וגם $\bar{x}$ נמצאים שניהם על מעגל מכוון אחד, אזי אין השמה מספקת לפסוק. נניח שיש קודקוד x כך שיש מסלול מ-x לעבר $\bar{x}$, וגם יש מסלול מ-$\bar{x}$ לעבר x.
  - אם $x = T$ אז צריך שגם $\bar{x} = T$ וזו סתירה!.
  - אם $\bar{x} = T$ אז צריך שגם $x = T$ וזו גם סתירה!

מ.ש.ל. - הקריטריון הכללי

המשך ההוכחה: לסיכום, בהינתן פסוק $2CNF$ נתאר מכונה/אלגוריתם דטרמיניסטי פולינומי: M_{2SAT} על קלט $< \phi >$ תבצע:

1. בונה את הגרף הנ"ל.
2. לכל $v \in V$ תבצע:
   a. מבצעת BFS (או DFS) מ-v.
   b. אם קיים מסלול מ-v אל $\bar{v}$, מבצעת גם BFS מ-$\bar{v}$, אם נמצא שהמסלול מגיע חזרה אל v - אז דחה.
3. קבל.

נכונות הבניה: נובעת ישירות נכונות הקריטריון שהוכחנו.

סיבוכיות זמן ריצה: סיבוכיות של DFS היא $O(|V| + |E|) = O(n + n^2)$ ונבצע את ה-DFS למשך $n = |V|$ פעמים. סה"כ נקבל $O(n^3)$ - זמן פולינומי.

∎

הסבר תכל'ס: נסתמך בשיטת הבדיקה הבאה: לפי בדיקת קיימות המסלול $x \to \bar{x}$ או/גם $\bar{x} \to x$ בגרף G, אנחנו יכולים להכריע האם ההשמה מספקת את ϕ או לא. קיום המסלול מצומת אחת לאחר יכולה להתבצע באלגוריתמים טריוויאלים לחיפוש בגרף כמו BFS או DFS. שני האלגוריתמים האלה הם בזמן פולינומי $O(V + E)$ ולכן אפשר להוכיח בצורה הזאת כי $2SAT \in P$.

בעיית כיסוי קודקודים היא NP-שלמה

<u>הגדרה</u>: קבוצת צמתים U בגרף לא מכוון $G = (V, E)$ נקראת כיסוי קודקודים, אם לכל קשת $(u, v) \in E(G)$ מתקיים ש: או $u \in U$ או ש: $v \in U$ או $v, u \in U$. בנוסף, צמתי U "מכסים" את כל קשתות הגרף.

$$VERTEX - COVER = \{< G, k > \mid G\ is\ an\ undirected\ graph\ that\ has\ a\ k - node\ vertex\ cover\}$$

<u>משפט</u>: $VC \in NPC$.

<u>הוכחה</u>: צריך להוכיח כי $VC \in NP$ ולאחר מכן להוכיח כי $VC \in NPH$ בעזרת רדוקציה $3SAT \leq_P VC$.

<u>חלק א'</u>: נוכיח כי $VC \in NP$. נתאר מ"ט א"ד לשפה VC.

המכונה N על קלט $< G, k >$ תבצע:

1. מנחשת קבוצת קודקודים בגודל k.
2. עוברת על כל הצלעות בגרף, אם מוצאת צלע שחלה בקודקוד שאינו בקבוצה - <u>דוחה</u>.
3. <u>מקבלת</u>.

נשים לב שלא ביקשנו כיסוי מינימלי ($\tau(G)$). יתכן ויהיו קודקודים זהים בקבוצה שמנחשים.

- המכונה עובדת בזמן פולינומי:
  - שלב 1 - כגודל הניחוש $O(k)$.
  - שלב 2 - מעבר על כל הצלעות בגרף ובודקת כל קודקוד בקבוצה $O(k \cdot |E|)$.

  סה"כ זמן ריצה $O(k \cdot |E|)$ - זמן פולינומי.
- נכונות הבניה: נראה כי $L(N) = VC$.
  - $< G, k > \in VC$ ← קיימת קבוצת k-קודקודים המכסים את כל הצלעות בגרף ← קיים ניחוש עבור קבוצה כזו בגודל k ← N בוחר בניחוש הנכון, עובר כל על הצלעות והבדיקה עברה בהצלחה ← N מקבל ← $< G, k > \in L(N)$.
  - $< G, k > \notin VC$ ← לא קיימת קבוצת k-קודקודים המכסים את כל הצלעות בגרף ← לכל ניחוש עבור קבוצה כזו בגודל k, N בודק ומוצא צלע שחלה בקודקוד שאינו בקבוצה ← N דוחה ← $< G, k > \notin L(N)$.

<u>חלק ב'</u>: נוכיח כי $VC \in NPH$ בעזרת רדוקציה $3SAT \leq_P VC$. לכל פסוק $\phi \in 3CNF$, יש לבנות זמן פולינומי בגודל של ϕ, גרף לא מכוון G ומספר טבעי k, כך שב-G יש VC בגודל k אם ורק אם ϕ ספיק.

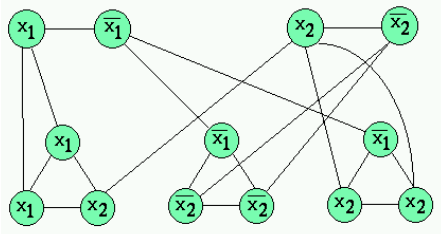
בהינתן פסוק $3CNF$ עם n משתנים ו-m פסוקיות, כאשר כל פסוקית φ_i היא פסוקית עם 3 ליטרלים וגם כל ליטרל הוא מהצורה $l_{ij} = x_k$ או $l_{ij} = \bar{x_k}$. נבנה את הגרף הבא $G = (V, E)$:

- קבוצת הקודקודים:

$$V = \{x_i \mid 1 \leq i \leq n\} \cup \{\bar{x_i} \mid 1 \leq i \leq n\} \cup \{l_{jr} \mid 1 \leq j \leq m,\ 1 \leq r \leq 3\}$$

צומת לכל משתנה — צומת שלילה לכל משתנה — לכל פסוקית j, ניצור שלושה צמתים

- קבוצת הצלעות E נבנה כך:
  - לכל משנה x_i נגדיר את הצלע $\{x_i, \bar{x}_i\}$.
  - לכל פסוקית j, ניצור משולש r (3 צלעות מחוברות) שקודקודיו הם הליטרלים של הפסוקית.
  - נחבר כל ליטרל במשולשים של הפסוקיות עם המשתנה המתאים לו בקשתות של המשתנים.



$$\phi(x_1, x_2) = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$

אפשר למשל לכתוב lx_{ij} כאשר j אינדקס המשולש.

הרדוקציה היא $f(<\phi>) =< G, k >$ כך ש-G הוא הגרף שתיארנו ו-$k = 2m + n$ (לכסות צלעות משתנים - n וגם לכסות משולשים $2m$ - כי לכל משולש, צריך 2 קודקודים בשביל לכסות אותו).

- <u>סיבוכיות זמן הריצה</u>: גודל הקלט הוא $O(m + n)$ וזהו גם סדר הגודל של הגרף שבנינו. כמות הצלעות היא בדיוק $3m + 3m + n$. אם אנחנו עובדים עם מטריצת שכנויות, אז זה ריבועי - עדיין פולינומי.
- <u>תקפות הפונקציה</u>:
  - נניח ש: $\phi \in 3SAT$ ותהי π השמה מספקת עבורה. נבנה כיסוי קודקודים ל-G שגודלו הוא בדיוק $2m + n$:
    - מהקשתות של המשתנים, ניקח כל זוג $\{x_i, \bar{x}_i\}$ את הליטרל שמקבל ערך T בהשמה π
    - לכל משולש של פסוקית, נבחר את כל קודקודי הליטרלים בפסוקית שלא הסתפקו. יש לכל היותר 2 כאלה לכל פסוקית, כי π היא השמה מספקת ולכן נותנת T ללפחות ליטרל אחד מכל פסוקית.

    גודל הקבוצה שתיארנו הוא $2m + n$ (לכל היותר).

    למה קבוצה זו מהווה כיסוי קודקודים? - נראה ששלושת סוגי הצלעות מכוסות:

    1. בין משתנה לשלילתו - בהשמה חוקים בדיוק אחד מהם מסתפק ולכן צלעות אלו מכוסות.
    2. בכל משולש יש 2 קודקודים בקבוצה - כי 2 קודקודים מכסים משולש.
    3. הסיבה שבגללה כל צלע בין החלק העליון (ממשתנה לשלילתו) לחלק התחתון (משולשי הפסוקיות) מכוסה מצריכה הסבר. נניח בשלילה שישנה צלע בין החלק העליון לחלק התחתון שאינה מכוסה - אם כך, שני קודקודי הצלע לא בקבוצת הקודקודים שנבחרו. הקודקוד בחלק העליון לא בקבוצה, כלומר הליטרל שהוא מייצג לא מסתפק, ולכן הפסוקית מטה לא מסתפקת ממנו. הקודקוד התחתון לא מיוצג בקבוצה ואם כך, יש כבר שני קודקודים בפסוקית שנבחרו (אחרת פשוט היינו מוסיפים אותם). אבל, לפי בניית הקבוצה הוספנו את כל הקודקודים מהמשולש שלא מסתפקים - סתירה. יש לכל היותר שני ליטרלים שלא מסתפקים בפסוקית ספיקה. סך-הכל, הקבוצה שבחרנו היא VC כי כל הצלעות מכוסות ע"י קודקודי הקבוצה.

- נניח ש: $< G, k > \in VC$, בגרף יש תת-קבוצה של קודקודים S שהיא כיסוי קודקודים בגודל $k = 2m + n$ בדיוק. נשתמש ב-S הנ"ל כדי למצוא השמה המספקת את הפסוק ϕ.

ראשית, כדי לכסות צלעות של משולש, צריך לבחור לפחות שני קודקודים, וכדי לכסות צלע בין משתנה לשלילתו צריך לפחות קודקוד אחד. מכיוון ש: $|S| = 2m + n$, נובע ש-S אכן מכיל לפחות קודקוד אחד מכל צלע של משתנה ושלילתו, ולפחות 2 קודקודים מכל משולש של פסוקית. נגדיר השמה מספקת ל-ϕ באופן הבא: ההשמה תתן T לליטרלים שנבחרו ל-S מתוך הצלעות בין משתנה ושלילתו, ו-F לאחרים. ההשמה הנ"ל חייבת להיות השמה מספקת - אחרת, נקבל שיש משולש שכל הצלעות שיוצאות ממנו לא מגיעות לקודקוד כלשהו ב-S, וזאת סתירה לכך ש-S היא כיסוי קודקודים של G. נרצה להראות שיש לפחות ליטרל אחד חיובי בכל פסוקית, מה שיגרור שהפסוק ספיק. נניח בשלילה שהיא לא מספקת את הפסוק - אם כך, יש לפחות פסוקית אחת שלא מסתפקת. מכאן שכל הליטרלים בה שליליים. בדיוק שני קודקודים מכל משולש נבחרו. נתבונן בקודקוד שלא נבחר בפסוקית זו. כדי שהליטרל בפסוקית שקודקוד זה מייצג לא יסתפק - הקודקוד שמייצג את ליטרל זה בחלק העליון לא נבחר אל S. מאחר וגם הקודקוד בפסוקית לא נבחר אליה, הצלע ביניהם לא מכוסה, סתירה.

∎

בעיית קבוצה בלתי תלויה היא NP-שלמה

<u>תזכורת</u>: קבוצה בלתי תלויה (ב"ת) או $INDEPENDENT - SET$ של צמתים בגרף לא מכוון G היא קבוצת צמתים שאין קשת בין כל שניים מהם.

$$IS = \{< G, k > \mid G \text{ is an undirected graph that has a } k - \text{node independent set}\}$$

<u>תרגיל</u>: הוכיחו: קבוצת צמתים U בגרף לא מכוון G היא כיסוי קודקודים ב-G, אם"ם $V\backslash U$ היא קבוצה ב"ת ב-G.

<u>פתרון</u>:

- כיוון א': נניח ש-S היא קבוצה בלתי תלויה ב-G, אזי אין צלעות המחברות <u>שני</u> קודקודים ב-S, כלומר, כל צלעות הגרף נוגעות בלכל היותר קודקוד אחד מ-S. אזי הקבוצה $V\backslash S$ נוגעת <u>בלפחות</u> קודקוד אחד מכל צלע בגרף, כלומר $V\backslash S$ היא כיסוי קודקודים.
- כיוון ב': נניח ש-$V\backslash S$ היא כיסוי קודקודים ב-G, אזי כל צלע ב-G היא בעלת <u>לפחות</u> קודקוד אחד בקבוצה $V\backslash S$. מכאן שאין צלעות שחלות על שני קודקודים מ-S, כלומר S היא קבוצה בלתי תלויה.

∎

משפט: $IS \in NPC$.

רעיון ההוכחה: (של הרדוקציה). נניח שקיים אלגוריתם יעיל (בזמן פולינומי) הפותר את בעיית הקבוצה הבלתי תלויה, אפשר פשוט להשתמש בה כדי להכריע האם ל-G יש כיסוי קודקודים בגודל לכל היותר k, על ידי השאלה האם G יש קבוצה ב"ת בגודל לפחות $n - k$.

הוכחה: צריך להוכיח כי $IS \in NP$ וגם $IS \in NPH$ ברדוקציה פולינומית $VC \leq_P IS$.

$IS \in NP$ הוכחנו כבר בפרק קודם. נוכיח כי $IS \in NPH$ ברדוקציה פולינומית $VC \leq_P IS$.

קלט: $< G, k > \in VC$ (לכל צלע בגרף G, לפחות אחד מקודקודיו שייך ל-S).

פלט: $< G, n - k > \in IS$ (לכל צלע בגרף G, יש לכל היותר קודקוד אחד ב-S).

פונקצית הרדוקציה: $f(< G, k >) =< G, n - k >$.

הסבר: אם קיימת ב-G קבוצה כיסוי קודקודים בגודל k, אז קבוצה ב"ת ב-G היא בגודל לפחות $n - k$.

הרדוקציה פולינומית: כדי לחשב את f, הרדוקציה רק מחשבת את מספר קודקודי הגרף $(n =)$ ומפחיתה מ-n ל-k. פעולות בזמן פולינומי.

תקפות הרדוקציה: תקפות הרדוקציה נובעת מהלמה הבאה:

למה: עבור גרף $G = (V, E)$ קבוצת קודקודים $S \subseteq V(G)$ היא בלתי תלויה ב-G אם"ם $V \backslash S$ היא כיסוי קודקודים של G.

■

בעיית הקליקה היא NP-שלמה

תזכורת: **קליקה,** $Clique$: בהינתן גרף $G = (V, E)$, קליקה ב-G היא תת קבוצה של קודקודים $S \subseteq V$ המקיימת שכל זוג קודקודים ב-S הם שכנים בגרף. (שיש צלע בין כל 2 צמתים בקבוצה).

$$CLIQUE = \{< G, k > | G \ is \ a \ graph \ that \ contains \ a \ clique \ of \ size \ k\}$$

משפט: $CLIQUE \in NPC$.

הוכחה: נוכיח כי $CLIQUE \in NP$ וגם $CLIQUE \in NPH$ בעזרת רדוקציה פולינומית $IS \leq_P CLIQUE$.

חלק א': נוכיח כי $CLIQUE \in NP$, הוכחנו בפרק הקודם.

חלק ב': נוכיח כי $CLIQUE \in NPH$ בעזרת רדוקציה פולינומית $IS \leq_P CLIQUE$, הוכחנו גם בפרק קודם.

מסקנה: $CLIQUE$ היא NP-שלמה.

תרגיל ממבחן

יהי $CliqueIS = \{< G, k_1, k_2 > \mid G\ is\ a\ graph\ with\ a\ Clique\ of\ size\ k_1\ OR\ an\ IS\ of\ size\ k_2\}$

סווגו את $CliqueIS$ ל-$NPC/NP/P$.

<u>פתרון</u>: $CliqueIS \in NPC$. צריך להראות $CliqueIS \in NP$ וגם $CliqueIS \in NPH$.

נוכיח כי $CliqueIS \in NP$, נציג הוכחה של חלק זה ב-2 שיטות.

1. נבנה N מ"ט א"ד פולינומית המכריעה את השפה $CliqueIS$.
2. נוכיח ביחס R_L.

<u>שיטה 1 - בניית מכונה</u>: המכונה N על הקלט $< G, k_1, k_2 >$ תבצע:

1. נחש קבוצת קודקודים S_1 בגודל k_1 וקבוצת קודקודים S_2 בגודל k_2.
2. בדוק האם הקבוצה S_1 היא קליקה, אם כן - <u>קבל</u>.
3. בדוק האם הקבוצה S_2 היא בלתי תלויה, אם כן - <u>קבל</u>.
4. אחרת, <u>דחה</u>.

<u>שיטה 1 - סיבוכיות זמן ריצה</u>: המכונה N היא פולינומית, הניחוש חסום פולינומית $2|V(G)| \geq (k_1 + k_2)$.

בנוסף, הבדיקה של S_1 לקליקה חסום ב-$O(k_1^2)$, הבדיקה של S_2 לב"ת חסום ב-$O(k_2^2)$.

סך הכל, פולינומי (ריבועי) בקלט.

<u>שיטה 1 - הוכחת נכונות הבניה</u>:

- $< G, k_1, k_2 > \in CliqueIS \leftarrow$ קיימת קליקה בגודל k_1 או קבוצה ב"ת בגודל $k_2 \leftarrow$ קיים ניחוש כזה והמכונה N תנחש נכון ותקבל $\leftarrow < G, k_1, k_2 > \in L(N)$.
- $< G, k_1, k_2 > \notin CliqueIS \leftarrow$ לא קיימת קליקה בגודל k_1 ולא קבוצה ב"ת בגודל $k_2 \leftarrow$ לכל ניחוש, המכונה N תדחה $\leftarrow < G, k_1, k_2 > \notin L(N)$.

<u>שיטה 2 - הגדרת היחס</u>: נגדיר את היחס $R_{CliqueIS}$. עבור $(x, y) \in R_{CliqueIS}$ נגדיר את $x =< G, k_1, k_2 >$ ונגדיר את העד: $y =< S_1, S_2 >$ כאשר $|S_2| = k_2\ ,\ |S_1| = k_1$.

<u>שיטה 2 - היחס חסום פולינומית</u>: העד y חסום פולינומית ב-$2|V(G)|$.

<u>שיטה 2 - היחס ניתן להכרעה פולינומית</u>: המכונה המוודאת V על קלט $(< G, k_1, k_2 >,\ < S_1, S_2 >)$ תוודא:

1. האם S_1 קליקה, אם כן - קבל.
2. האם S_2 ב"ת, אם כן - קבל.
3. אחרת - דחה.

<u>שיטה 2 - המכונה המוודאת חסום פולינומי</u>: הבדיקה של S_1 לקליקה חסום ב-$O(k_1^2)$, הבדיקה של S_2 לב"ת חסום ב-$O(k_2^2)$. סך הכל, פולינומי (ריבועי) בקלט.

שיטה 2 - נכונות הבניה של המכונה המוודאת:

- $< G, k_1, k_2 > \in CliqueIS \leftarrow$ קיימת קליקה בגודל k_1 או קבוצה ב"ת בגודל $k_2 \leftarrow$ המכונה המוודאת V תקבל $\leftarrow (< G, k_1, k_2 >, \ < S_1, S_2 >) \in R_{CliqueIS}$.
- $< G, k_1, k_2 > \notin CliqueIS \leftarrow$ לא קיימת קליקה בגודל k_1 ולא קבוצה ב"ת בגודל $k_2 \leftarrow$ לכל עד, המכונה המוודאת V תדחה $\leftarrow (< G, k_1, k_2 >, \ < S_1, S_2 >) \notin R_{CliqueIS}$.

סיכום שלב 1: הוכחנו $CliqueIS \in NP$ ב-2 שיטות: 1) בניית מכונה א"ד. 2) יחס R_L.

שלב 2: נוכיח כעת $CliqueIS \in NPH$ בעזרת רדוקציה פולינומית $CLIQUE \leq_P CliqueIS$.

פונקצית הרדוקציה: $f(< G, k >) = (< G', k_1, k_2 >)$.

רעיון הרדוקציה: ניקח גרף מ-$CLIQUE$, ונרצה לטפל רק בקליקה של $CliqueIS$. מאחר ותנאי השפה של $CliqueIS$ הוא קליקה "או" קבוצה ב"ת, יתכן שנגיד ש-$< G, k > \notin CLIQUE$ ואז צריך שלא יהיה קליקה בגרף G', אבל מי אמר לנו שאין שם במקרה קבוצה ב"ת? לכן, אנחנו נאלץ ב-G' שתמיד **לא** תהיה לנו קבוצה ב"ת, וזה בסדר. איך נאלץ? נגדיר את k_2 להיות יותר ממספר הקודקודים בגרף G' - וזה לעולם לא יתקיים.

תיאור הרדוקציה: בגרף G' נגדיר $k_1 = k$. כדי לוודא שאין קבוצה ב"ת בגודל k_2, נגדיר $k_2 = |V(G')| + 1$.

הפונקציה ניתנת לחישוב ומלאה: מתקיים בזמן פולינומי כיוון שביצענו רק הגדרות (השמות).

הפונקציה תקפה:

- $< G, k > \in CLIQUE \leftarrow$ קיים בגרף G קליקה בגודל $k \leftarrow$ קיים בגרף G' קליקה בגודל $k_1 \leftarrow < G', k_1, k_2 > \in CliqueIS$.
- $< G, k > \notin CLIQUE \leftarrow$ לא קיים בגרף G קליקה בגודל $k \leftarrow$ לא קיים בגרף G' קליקה בגודל k_1, בנוסף, לא קיימת קבוצה ב"ת כי לא יתכן שקיימת קבוצה ב"ת בת $|V(G)| + 1$ קודקודים $\leftarrow < G', k_1, k_2 > \notin CliqueIS$.

סיכום: הוכחנו $CliqueIS \in NP$ בעזרת בנית מ"ט א"ד פולינומית וגם בדרך אחרת עם יחס R_L.

בשלב השני, הוכחנו $CliqueIS \in NPH$ בעזרת הרדוקציה הפולינומית $CLIQUE \leq_P CliqueIS$.

ומכאן $CliqueIS \in NPC$.

∎

טיפ למבחן - שימוש בגאדג'טים (Gadgets)

בהרצאות לא נתנו שם לשיטה הזאת אבל בעצם השתמשנו בה כמה פעמים, זו שיטה שיכולה לעזור במבחן במיוחד בפתרון של בעיות חדשות או דומות. גאדג'ט היא קבוצת קודקודים/צלעות שמייצגת גוף/חלק מתוך פסוקית. למשל, ברדוקציה שעשינו $3SAT \leq_P VC$ יצרנו 2 גאדג'טים:

1. קבוצה 1 - הליטרלים.
2. קבוצה 2 - הפסוקיות (המשולשים).

השליטה בגאדג'טים עזרה לנו לשלוט בעניין שאם קיימת השמה מספקת ב-ϕ אז קיימת קבוצת כיסוי קודקודים בגרף בגודל k. נציג כעת את בעיית מסלול המילטון ונראה שהיא NP-שלמה בעזרת רדוקציה עם שימוש בגאדג'טים.

בעיית מסלול המילטון היא NP-שלמה

<u>תזכורת</u>: מסלול המילטון בגרף מכוון G הוא מסלול פשוט (ללא מעגלים) שמבקר בכל צומת פעם אחת ויחידה.

$$HAMPATH = \{< G, s, t > \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$$

<u>משפט</u>: $HAMPATH \in NPC$.

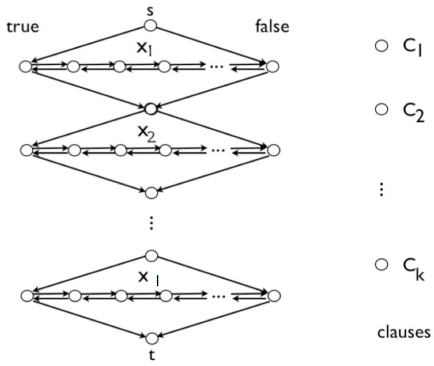
<u>הוכחה</u>: נוכיח כי $HAMPATH \in NP$ וגם $HAMPATH \in NPH$ בעזרת רדוקציה פולינומית $3SAT \leq_P HAMPATH$.

<u>חלק א'</u>: הוכחנו כבר בפרק קודם כי $HAMPATH \in NP$.

<u>חלק ב'</u>: נוכיח כי $3SAT \leq_P HAMPATH$.

אם ϕ פסוק ספיק אז קיים מסלול אוילר בגרף G.

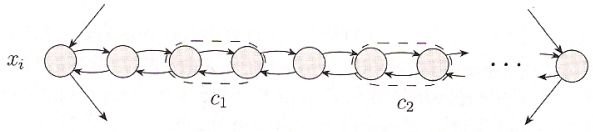
נשתמש בשני גאדג'טים:



- t משתנים - לכל משתנה x_i קיים גרף בצורת יהלום שממשיך לגרף יהלום אחר של המשתנה x_{i+1}.
- k פסוקיות - לכל פסוקית, קיים קודקוד בודד ללא שכנים.

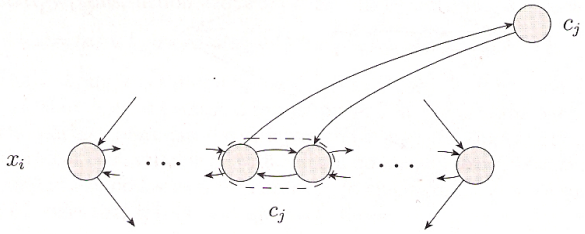
תת הגרף הפנימי בכל יהלום (השרשרת) מכיל $3k + 3$ קודקודים:

- 3 לכל פסוקית.
- קודקוד כדי לבודד אותם מ-2 קודקודי הליטרל.
- שני קודקודים נוספים לכל צד עבור הליטרלים $x_i, \bar{x}_i$.

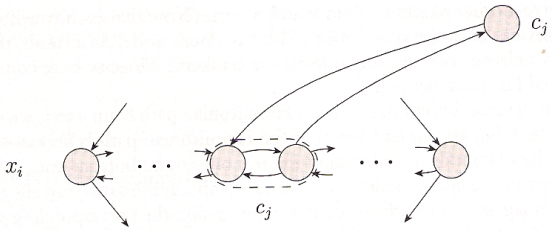


<u>טיפול בצלעות</u>:

אם x_i מופיע בפסוקית c_j, נוסיף 2 צלעות מהקבוצה ה-j-ית בשרשרת לקודקוד הפסוקית ה-j ביהלום (המשתנה) ה-x_i.

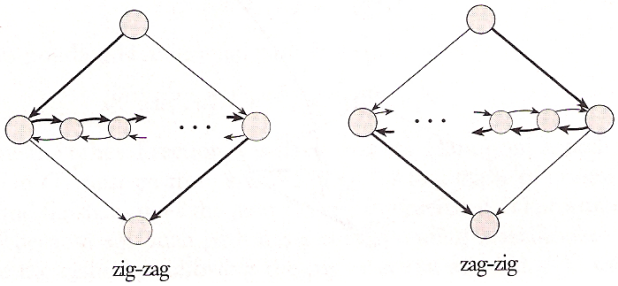


אם $\overline{x_i}$ מופיע בפסוקית c_j, נוסיף 2 צלעות מהקבוצה ה-j-ית בשרשרת לקודקוד הפסוקית ה-j ביהלום (המשתנה) ה-x_i, אבל ברוורס.



<u>תקפות</u>:

- $3SAT \in <\phi> \leftarrow$ ספיק $\phi \leftarrow$ אם נתעלם לרגע מקודקודי הפסוקיות, נשים לב שביהלומים קיים מסלול המילטוני: נתחיל מ-s ונטייל לאורך כל יהלום עד שנגיע ל-t. $\leftarrow$ ביהלום ה-i, הוא מטייל משמאל לימין או מימין לשמאל, תלוי בערך של x_i (כלומר T/F). $\leftarrow$ קודקודי הפסוקיות יכולים להשתלב במסלול בעזרת בניית 2 הצלעות שסיפקנו להם מהיהלומים. $\leftarrow$ מצד אחד, אם $x_i = T$ ונמצא בפסוקית c_j, אז אפשר לעשות מסלול הלוך וחזור (אל הצומת c_j וחזרה לשרשרת בכיוון הנכון). מצד שני, אם $\bar{x}_i = T$ ונמצא בפסוקית c_j, אז אפשר לעשות מסלול הלוך וחזור (אל הצומת c_j וחזרה לשרשרת בכיוון הנכון) אבל בכיוון ההפוך. $\leftarrow HAMPATH \in <G>$



- $HAMPATH \in <G> \leftarrow$ קיים מסלול אוילר בגרף $G \leftarrow$ אם המסלול נורמלי, אז הוא מטייל מ-s בזיגזגים לאורך היהלומים $\leftarrow$ קיימת השמה מספקת עבור ϕ. $\leftarrow 3SAT \in <\phi>$.

∎

בעיית SUBSET-SUM היא NP-שלמה

<u>תזכורת</u>: הקלט הוא קבוצה S של מספרים ומספר t. השאלה: האם יש ל-S תת-קבוצה שהסכום שלה בדיוק t?

$$SUBSET-SUM=\{<S,t>\mid S=\{x_1,..,x_n\},\ \exists\{y_1,..,y_k\}\subseteq S,\ \Sigma y_i=t\}$$

<u>משפט</u>: $SUBSET-SUM\in NPC$.

<u>הוכחה</u>: נוכיח כי $SUBSET-SUM\in NP$ וגם $SUBSET-SUM\in NPH$ בעזרת רדוקציה פולינומית $3SAT\leq_P SUBSET-SUM$.

<u>חלק א'</u>: הוכחנו כבר בפרק קודם כי $SUBSET-SUM\in NP$.

<u>חלק ב'</u>: נוכיח כי $3SAT\leq_P SUBSET-SUM$.

מפסוק $\phi\in 3CNF$ נבנה קבוצת מספרים טבעיים S ומספר טבעי t באופן הבא:

- כל המספרים יהיו בייצוג עשרוני, כל המספרים יהיו באורך $O(n+m)$ (אם המספר יהיה קצר יותר אז נרפד אותו באפסים). נסמן n - מספר המשתנים ונסמן m - מספר הפסוקיות.
- נייצר את המספרים בקבוצה S כך שספרותיהם יהיו רק $0, 1, 2$.
- כל ספרה במספרים שנייצר תייצג ערך T לעמודה בה היא נמצאת.
- לכל משתנה x_i בפסוק נייצר 2 מספרים y_i, z_i.
- לכל פסוקית c_j נייצר 2 מספרים $g_j,\ h_j$.
- המספרים יווצרו על ידי הטבלה הבאה:

<u>תיאור הטבלה ושיטתה</u>:

- באגף הימני של המשתנים:
  - המשתנה $x_i=T$ אם"ם בשורה y_i ובעמודה x_i יהיה 1. ביתר השורה עבור יתר המשתנים יהיה 0.
  - המשתנה $x_i=F$ אם"ם בשורה z_i ובעמודה x_i יהיה 1. ביתר השורה עבור יתר המשתנים יהיה 0.
- באגף השמאלי של הפסוקיות:
  - בפסוקית c_i קיים x_i אם"ם בשורה y_i ובעמודה c_i יהיה 1. ביתר השורה עבור יתר הפסוקיות יהיה 0.
  - בפסוקית c_i קיים $\bar{x_i}$ אם"ם בשורה z_i ובעמודה x_i יהיה 1. ביתר השורה עבור יתר הפסוקיות יהיה 0.

באגף הימני של המשתנים: נרפד ב-0.

באגף השמאלי של הפסוקיות: בשורות $g_j,\ h_j$ בעמודה ה-c_j יהיה 1 וביתר הפסוקיות יהיה 0.

בניית המספר t:

- באגף הימני של המשתנים: יהיה 1 בכל עמודה - זאת על מנת להבטיח שלכל משתנה תהיה הצבה אחת בדיוק שהיא 0 או 1.
- באגף השמאלי של הפסוקיות: יהיה 3 בכל עמודה. בצורה זו לכל פסוק.

	x_1	x_2	....	x_l	c_1	c_2	....	c_k
y_1	1	0		0	1	0		0
z_1	1	0		0	0	1		0
y_2		1		0	0	0		1
z_2		1		0	0	0		0
...								
y_l				1	0	0		0
z_l				1	0	0		0
g_1					1	0		0
h_1					1	0		0
g_2						1		0
h_2						1		0
...								
g_k								1
h_k								1
t	1	1	...	1	3	3	...	3

תרגיל 1 (עוד לא סיימנו להוכיח): איזו ספרה תהיה במספר t במקום של כל פסוקית?. תשובה: 3.

תרגיל 2 (עוד לא סיימנו להוכיח): בנו את קבוצת המספרים S ואת המספר t המתאימים לפסוק הבא:

$$(x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee w) \wedge (y \vee z \vee \bar{w})$$

תרגיל 3 (עוד לא סיימנו להוכיח): הוכיחו שהיא ניתנת לחישוב בזמן פולינומי. הוכיחו שהרדוקציה תקפה.

פתרון 3: יש לנו בדיוק $m + n$ משתנים, כל אחד מהם בדיוק באורך $m + n$. סה"כ $O(n \times m)$ - פולינומי.

מסקנה: $SUBSET - SUM$ היא NP-שלמה.

בעיות NP-שלמות נוספות

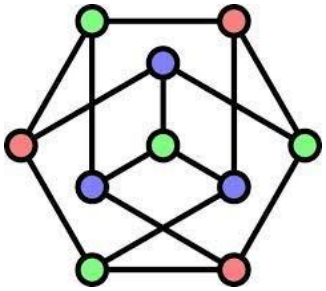
1. גרפים לא מכוונים שיש להם צביעה חוקית בשלושה צבעים נעשה ברדוקציה של $3SAT$.
2. גרפים לא מכוונים שיש להם מסלול המילטון מ-s ל-t נעשה ברדוקציה על $HAMPATH$.
3. גרסה נוספת של מסלול המילטון:

$$HAMPATH' = \{< G > \mid G \text{ is an undirected graph with a Hamiltonian path}\}$$

4. מעגל המילטון:

$$HAMCYCLE = \{< G > \mid G \text{ is an undirected graph with a Hamiltonian cycle}\}$$

בעיית גרף 3 צביע היא NP-שלמה



הגדרה: בתורת הגרפים, גרף n-צביע הוא גרף שאפשר לצבוע את הקודקודים שלו ב-n צבעים, כך ששני קודקודים סמוכים אינם צבועים באותו צבע. עבור גרף G , מסמנים ב-$\chi(G)$ את המספר הקטן ביותר של צבעים הדרוש לצביעת הקודקודים שלו. מספר זה נקרא מספר הצביעה (או המספר הכרומטי) של הגרף.עבור $k > 2$, ההכרעה האם $\chi(G) = k$ היא בעיה NP שלמה. לגרף יש

מספר כרומטי 2 אם ורק אם הוא דו-צדדי, ותכונה זו ניתנת לזיהוי בזמן ליניארי במספר הקשתות.

<u>משפט</u>: $3COLOR \in NPC$.

<u>הוכחה</u>: נוכיח כי $3COLOR \in NP$ וגם $3COLOR \in NPH$ ברדוקציה פולינומית $3SAT \leq_P 3COLOR$.

הוכחנו כבר בפרק קודם ש: $3COLOR \in NP$, בנינו מכונה א"ד פולינומית שמנחשת צבע עבור כל צומת בגרף (צבע אחד מתוך ה-3) ואז בודקת בכל צלע אם שני הקצוות שלו בעלות צבע שונה.

כעת נוכיח ברדוקציה פולינומית $3SAT \leq_P 3COLOR$. בהינתן פסוק $\phi \in 3CNF$, נרצה לבנות גרף שיהיה 3-צביע אם"ם הפסוק מסתפק. הפסוק שנעסוק בו הוא $\phi(x_1,\ldots,x_n)$ בעל m פסוקיות.

הגרף $G = (V, E)$ יבנה באופן הבא:

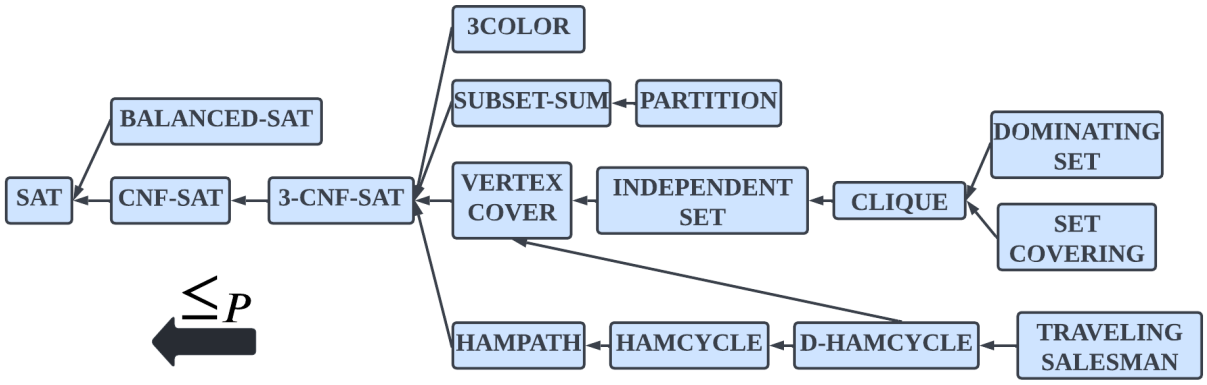
1. ניצור 3 קודקודים T, F, B ונחבר אותם.
2. לכל ליטרל בפסוק, ניצור קודקוד ונחבר אותו ל-B.
3. נחבר בין כל משתנה לשלילתו.
4. לכל שלושה ליטרלים המהווים פסוקית, נוסיף 'תוספת' בעלת 6 קודקודים ו-13 צלעות.

<u>סיבוכיות זמן הריצה</u>: גודל הקלט הוא $O(m + n)$, בגרף יש $O(3 + n + 3m) = O(n + m)$ קודקודים ו-$O(3n + 13n) = O(n + m)$ צלעות ולכן סה"כ פולינומי בגודל הקלט.

<u>תקפות הרדוקציה</u>:

- נניח ש-ϕ ספיק ונרצה להוכיח שהגרף הוא צביע. לכל פסוקית יש לפחות ליטרל אחד חיובי. ניקח ליטרל אחד חיובי כזה ובהתאמה **לאיור משמאל?**, נצבע אותו בירוק, את הקודקוד תחתיו באדום ואת הקודקוד תחתיו בכחול. את יתר השורה העליונה נבצע באדום. את יתר השורה האמצעית נצבע בכחול. ואת יתר הקודקודים הנותרים בשורה התחתונה נצבע בהתאמה לאיור. סה"כ לכל 'תוספת' של פסוקית תהיה השמה מספקת. ההשמות מכבדות זו את זה.
- נניח שישנו גרף בצורה הזו ונרצה לבנות ל-ϕ השמה מספקת. תחילה נשים לב שההשמות חוקיות- לכל ליטרל יהיה בדיוק ערך אחד)ירוק או אדום(, משום שהם מחוברים ל- B . בנוסף, לכל משתנה תהיה השמה אחת בדיוק שהפוכה לשלילה שלו- משום שבין כל משתנה ושלילתו יש צלע. ניקח צביעה כלשהי ונקבע השמה באופן הבא - כל ליטרל שבירוק יהיה T . נותר להראות שכל פסוקית מסתפקת. נניח בשלילה שישנה פסוקית שלא מסתפקת, אם כך כל הקודקודי המייצגים את הליטרלים שלה הם באדום. אם כך, כל קודקודי האמצע הם כחולים בהכרח, משום שלכל אחד מהם שכן כחול ושכן ירוק. סתירה.

סיכום מפת עולם לרדוקציות NP-שלמות



תרגיל 1 - הוכחת שפה ב-NPC בעזרת הרכבת פונקציות רדוקציה - תרגול 7

נגדיר את השפה $COLOR = \{(< G >, k) |\ G\ can\ be\ colored\ with\ k\ colors\}$.

הראו רדוקציה $CNF - SAT \leq_p COLOR$.

הסבר: אנחנו עושים רדוקציה $CNF - SAT \leq_p COLOR$, נעזר ברדוקציות שכבר הוכחנו בקורס ככלי לפתרון הבעיה שלנו. נעביר לצורת $3SAT$ בעזרת $CNF - SAT \leq_p 3 - CNF - SAT$, לאחר מכן, נעביר לצורת $3COLOR$ בעזרת $3 - CNF - SAT \leq_p 3COLOR$ ואז נבחר $k = 3$ שיתאים לבעיה $COLOR$.

פונקצית הרדוקציה שלנו היא $f(< \phi >) = (g(h(\phi)),\ 3)$. כאשר $=< G >= g(h(\phi))$ כלשהו.

פתרון:

נסמן ב-h את פונקצית הרדקוציה של $CNF - SAT \leq_p 3 - CNF - SAT$.

נסמן ב-g את פונקצית הרדקוציה של $3 - CNF - SAT \leq_p 3COLOR$.

פונקצית הרדוקציה: $f(< \phi >) = (g(h(\phi)),\ 3)$.

תקפות:

- $< \phi >\in CNF - SAT \leftarrow h(< \phi >) =< \phi' >\in 3 - CNF - SAT$, מהתקפות של $h \leftarrow$ $g(h(< \phi >)) = g(< \phi' >) =< G >\in 3COLOR$, מהתקפות של $g \leftarrow$ $(g(h(< \phi >)), 3) \in COLOR$.
- $(g(h(< \phi >)), 3) \in COLOR \leftarrow g(h(< \phi >)) \in 3COLOR \leftarrow$ לפי התקפות של g נקבל כי $h(< \phi >) \in 3 - CNF - SAT \leftarrow$ לפי התקפות של h נקבל כי $< \phi >\in CNF - SAT$.

∎

תרגיל 2 - מסלול המילטוני או קליקה - תרגול 7

נתונה השפה: $HamOrClique = \{(G,k) | \ G\ has\ a\ Hamiltonian\ path\ or\ a\ Clique\ of\ size\ k\}$.

האם $HamOrClique \in P$ או $HamOrClique \in NPC$?

<u>פתרון</u>: $HamOrClique \in NPC$.

תחילה נוכיח כי $HamOrClique \in NP$ בעזרת בנית מ"ט א"ד N המכריעה את השפה $HamOrClique$ בזמן פולינומי. המכונה N על קלט $< G,k >$ כאשר G הוא גרף ו-k מספר טבעי וחיובי, תבצע:

1. נחש קבוצת קודקודים S_1 ונחש סדרת קודקודים S_2.
2. בדוק שכל קודקוד ב-S_1 מופיע פעם אחת בלבד וגם בדוק שכל הקודקודים בגרף G נמצאים ב-S_2 ומופיעים פעם אחת בלבד. אם שתי הבדיקות נכשלו - <u>דחה</u>.
3. עבור כל זוג קודקודים $v,u \in S_1$ כאשר $u \neq v$:
   a. בדוק בעזרת מטריצת שכנויות שקיימת צלע בין u ל-v.
4. אם שלב 3 הצליח, <u>קבל</u>.
5. עבור כל קודקוד $v_i \in S_2$ כאשר $i \in [1, n-1]$ בצע:
   a. בדוק שקיים צלע בינו לבין v_{i+1}, אחרת <u>דחה</u>.
   b. המשך לקודקוד הבא v_{i+1}.
6. <u>קבל</u>.

<u>סיבוכיות זמן ריצה</u>: גודל הניחוש הוא $O(|S_1| + |S_2|) = O(n)$. בדיקת כפילויות בשימוש מטריצת שכנויות $O(n^2)$. בדיקת הקליקה $O(n^2)$ ובדיקת המסלול ההמילטוני $O(n^2)$ (בעזרת מטריצת שכנויות) סה"כ פולינומי.

<u>הוכחת נכונות הבניה</u>:

- $< G,k > \in HamOrClique$ ← קיים ב-G קליקה בגודל k או מסלול המילטוני ← קיים ניחוש של קבוצת קודקודים S_1 המתארת קליקה או סדרת קודקודים S_2 המתארת מסלול המילטוני ← N מנחש נכון, מבצע בדיקה שכל הקודקודים ב-S_1 מופיעים פעם אחת וגם שכל $V(G)$ נמצא ב-S_2 ופעם אחת : ←
  - בודק ב-S_1 שכל זוג קודקודים קיימת צלע.
  - או שבודק ב-S_2 שקיימת צלע בין כל 2 קודקודים עוקבים בסדרה.

  באחד מ-2 המקרים N מקבל ← $< G,k > \in L(N)$.
- $< G,k > \notin HamOrClique$ ← לא קיים ב-G קליקה בגודל k וגם לא מסלול המילטוני ← לכל ניחוש של N:
  - קיים קודקוד המופיע לפחות פעמיים בקבוצה S_1.
  - הקבוצה S_2 לא מכילה את כל הקודקודים ב-G.
  - קיים קודקוד המופיע לפחות פעמיים בקבוצה S_2.

- בקבוצה S_1 קיימים זוג קודקודים שונים שאין ביניהם צלע.
- בקבוצה S_2 קיימים זוג קודקודים עוקבים מהסדרה שאין ביניהם צלע.

כל אחת מהתרחישים האפשריים האלה N דוחה. $\leftarrow L(N) \not\ni < G,k >$.

כעת נותר להוכיח כי $HamOrClique \in NPH$. נראה רדוקציה $CLIQUE \leq_p HamOrClique$.

<u>פונקצית הרדוקציה</u>: $f(< G,k >) = (< G',k >)$.

<u>הסבר</u>: אנחנו נרצה שאם ב-G לא קיימת קליקה אז ב-G' לא קיימת קליקה **וגם** לא קיים מסלול המילטוני.
כדי להבטיח את זה, נאלץ את G' לא להכיל אף פעם מסלול המילטוני - זאת בעזרת הוספת קודקוד חדש ובודד לגרף G'.

<u>הפונקציה מלאה וניתנת לחישוב בזמן פולינומיאלי</u>: בניית G' על ידי העתקת גרף G עם תוספת קודקוד בודד אחד בזמן קבוע $O(1)$.

<u>הפונקציה תקפה</u>:

- $< G,k > \in CLIQUE \leftarrow$ קיים בגרף G קליקה בגודל $k \leftarrow$ גרף G' הוא העתק של G עם קודקוד מבודד נוסף שלא פוגע בהגדרת הקליקה $\leftarrow$ קיים ב-G' קליקה בגודל $k \leftarrow$ $< G',k > \in HamOrClique$.
- $< G',k > \in HamOrClique \leftarrow$ קיים בגרף G' קליקה בגודל k או מסלול המילטוני $\leftarrow$ לא יתכן כי קיים מסלול המילטוני כי קיים קודקוד מבודד ב-G' ולכן קיים רק קליקה בגודל $k \leftarrow$ מאחר ו-G' הוא העתק של G אך עם קודקוד מבודד נוסף, בהסרת הקודקוד הזה נקבל את $G \leftarrow$ קיים ל-G קליקה מגודל $k \leftarrow < G,k > \in CLIQUE$.

∎

תרגיל 3 - מסלול המילטוני <u>וגם</u> קליקה - תרגול 7

נתונה השפה: $HamAndClique = \{(G,k) |\ G\ has\ a\ Hamiltonian\ path\ and\ a\ Clique\ of\ size\ k\}$.
האם $HamAndClique \in P$ או $HamAndClique \in NPC$?

<u>פתרון</u>: $HamAndClique \in NPC$.

נוכיח כי $HamAndClique \in NP$. אותה הוכחה כמעט כמו תרגיל 2 רק שנרצה ש-2 הבדיקות עבור S_1 וגם S_2 יעבדו. נוכיח כי $HamAndClique \in NPH$.

<u>אבחנה</u>: קליקה S בגודל 1 (הקודקוד עצמו) בגרף G מחוברת לכל קודקוד אחר (עצמו) בקבוצה S.

נראה רדוקציה $HAMPATH \leq_p HamAndClique$.

<u>פונקצית הרדוקציה</u>: $f(< G >) = (< G >, 1)$.

<u>תקפות</u>:

- $< G > \in HAMPATH \leftarrow$ יש ב-G מסלול המילטוני $\leftarrow$ יש בגרף גם קליקה בגודל 1 $\leftarrow (< G >, 1) \in HamAndClique$.

- $HamAndClique \in (< G >, 1)$ ← יש ב-G מסלול המילטוני וגם קליקה בגודל 1 ← $< G > \in HAMPATH$.

∎

תרגיל 4 - תרגול 7

נתונה השפה:

$$L_{\log n} = \{\phi | \ \phi \ has\ a\ satisfying\ assignment\ \pi,\ and\ \exists i\ s.t.\ \forall_{j \notin [i+1, i+2, \ldots, i+\log n]} \pi(x_j) = T\}$$

האם השפה ב-P או ב-NPC?

<u>בשפת בני אנוש</u>: קיימת ל-ϕ השמה מספקת, כך שכל המשתנים מקבלים T חוץ מרצף של $\log(n)$ משתנים שיכולים לקבל מה שבא להם T או F.

<u>הסבר לפתרון</u>: אם יש לנו $\log_2(n)$ משתנים שאפשר לשחק איתם איך שבא לנו, אז יש לנו $2^{\log_2(n)}$ אופציות, כי לכל אחד מה-$\log_2(n)$-ים יש 2 אפשרויות: $\{T, F\}$. בדרך כלל אמרנו שאנחנו לא יכולים לבדוק השמה מספקת כי יש לנו לעבור על 2^n אופציות שזה אקספוננציאלי, אבל עכשיו יש לנו $2^{\log_2(n)}$ אופציות, ולפי חוקי לוג $n = a^{\log_a(n)}$ כלומר יש לנו n אופציות וזה אכן פולינומי!. לכן, $L_{\log n} \in P$.

<u>פתרון</u>: $L_{\log n} \in P$. נבנה מ"ט דטרמיניסטית מכריעה עבור $L_{\log n}$.

M על קלט ϕ תבצע:

- עבור $0 \leq i \leq n - \log(n)$:
  - עבור כל השמה של המשתנים בסדר רציף $x_{i+1}, x_{i+2}, \ldots, x_{i+\log(n)}$:
    - אם ההשמה הנוכחית + כל שאר המשתנים קיבלו T - <u>קבל</u>.
- <u>דחה</u>.

<u>סיבוכיות זמן ריצה</u>: עבור כל i, מספר ההשמות האפשריות הוא $2^{\log(n)} = n$ ← סה"כ נבדוק $O(n \cdot n)$ השמות ← בדיקה של השמה אחת $O(n)$ ← זמן הריצה של המכונה הוא $O(n^3)$ - <u>פולינומי</u>.

<u>נכונות הבניה</u>: נובעת ישירות מהבניה.

∎

תרגיל 5 - EXACT-SUBSET-SUM - תרגול 7

<u>תזכורת</u>: $SUBSET - SUM = \{< S, t > | \ S = \{x_1, \ldots, x_n\}, \exists \{y_1, \ldots, y_k\} \subseteq \{x_1, \ldots, x_n\}\ s.t.\ \Sigma y_i = t\}$ הוכחנו כי $SUBSET - SUM \in NPC$.

<u>הגדרה</u>: $EXACT - SUBSET - SUM = \{(x_1, \ldots, x_n), k, m \mid \exists I \subseteq [n]\ s.t.\ \Sigma_{i \in I} x_i = k \ \wedge\ |I| = m\}$

<u>בעברית</u>: בהינתן קבוצת מספרים $S = (x_1, \ldots, x_n)$, ומספרים טבעיים k, m. קיימת תת קבוצה $V \subseteq S$ כך שסכומה הוא k וגם $|V| = m$.

<u>תרגיל</u>: הוכיחו כי $EXACT - SUBSET - SUM \in NPC$.

<u>פתרון</u>: תחילה נוכיח כי $EXACT - SUBSET - SUM \in NP$.

נבנה מ"ט א"ד N המכריעה את השפה $EXACT - SUBSET - SUM$.

המכונה N על קלט $((x_1,..,x_n),k,m)$ כאשר $(x_1,..,x_n)$ היא קבוצת מספרים וגם k,m מספרים טבעיים, תבצע:

1. נחש קבוצה V של m מספרים.
2. בדוק שכל המספרים ב-V נמצאים ב-$(x_1,..,x_n)$, - אחרת, <u>דחה</u>.
3. בדוק שסכום הקבוצה V הוא k - אחרת, <u>דחה</u>.
4. <u>קבל</u>.

<u>המכונה ניתנת לחישוב בזמן פולינומי</u>: גודל הניחוש $O(m)$, בדיקת קיימות $O(n^2)$, בדיקת סכימה $O(m)$.

סה"כ $O(n^2)$ - <u>פולינומי</u>.

<u>נכונות הבניה</u>:

- $((x_1,..,x_n),k,m) \in EXACT - SUBSET - SUM$ ← קיימת תת קבוצה $V \subseteq S = (x_1,..,x_n)$ כך שסכומה הוא k וגם $|V| = m$ ← קיים ניחוש של תת קבוצה V סכומה הוא k וגודלה הוא $m \leftarrow N$ תנחש נכון את הקבוצה V, תבדוק שכל המספרים ב-V נמצאים גם ב-$S \leftarrow$ תבדוק שסכום הקבוצה V הוא $k \leftarrow N$ תצליח בכל הבדיקות ותקבל $((x_1,..,x_n),k,m) \in L(N)$.
- $((x_1,..,x_n),k,m) \notin EXACT - SUBSET - SUM$ ← לא קיימת תת קבוצה $V \subseteq S = (x_1,..,x_n)$ כך שסכומה הוא k וגם $|V| = m$ ← לכל ניחוש של V, המכונה N תבדוק:
  - שכל המספרים ב-V נמצאים גם ב-S
  - תבדוק שסכום הקבוצה V הוא k

  ← N תדחה באחת הבדיקות לעיל ← $((x_1,..,x_n),k,m) \notin L(N)$.

כעת נותר להוכיח כי $EXACT - SUBSET - SUM \in NPH$.

נראה רדוקציה $SUBSET - SUM \leq_p EXACT - SUBSET - SUM$.

<u>פונקצית הרדוקציה</u>: $f(< (x_1,....,x_n),k >) = (< (x_1,....,x_n,0,0,....,0,0),k,n >)$.

<u>הסבר</u>: אנחנו נרפד n אפסים בקבוצה הימנית.

<u>תקפות</u>:

- $< (x_1,....,x_n),k > \in SUBSET - SUM$ ← יש תת קבוצה המסתכמת ל-k, בגודל r כך ש: $r \leq n$ ← אם נוסיף לקבוצה r אפסים, נקבל קבוצה שעדיין מסתכמת ל-k וגודלה הוא n. ← $< (x_1,....,x_n,0,0,....,0,0),k,n > \in EXACT - SUBSET - SUM$.
- $< (x_1,....,x_n,0,0,....,0,0),k,n > \in EXACT - SUBSET - SUM$ ← יש תת קבוצה המסתכמת ל-k וגודלה הוא $n \leftarrow$ אותה קבוצה בלי האפסים עדיין מסתכמת ל-$k \leftarrow$ זו תהיה תת הקבוצה $(x_1,..,x_n) \leftarrow < (x_1,....,x_n),k > \in SUBSET - SUM$.

∎

תרגיל 6 - תרגול 7

נגדיר שפה:

$$L_{1/8} = \{\phi | \ \phi \ is\ 3CNF\ and\ \exists\ a\ satisfying\ assignment\ to\ \phi,$$
$$and\ at\ least\ \frac{1}{8}\ of\ the\ possible\ assignments\ are\ not\ satisfying\}$$

הוכיחו ש: $L_{1/8} \in NPH$.

<u>הוכחה</u>: נוכיח ברדוקציה $3 - CNF - SAT \leq_p L_{1/8}$.

<u>הסבר</u>: עד עכשיו תמיד בשאלות עם ϕ היינו לוקחים את הפסוק הקיים, ומוסיפים לו עוד פסוקית ב-AND (או כמה כאלה) שתעזור לנו להמיר לבעיה הנוכחית. אנחנו נרצה שלפחות $\frac{1}{8}$ מההשמות שלנו יהיו לא מספקות, לכן נרצה שהפסוקית החדשה תשלוט בקיום הדרישה הזאת. לפסוקית הנוכחית, נוסיף פסוקית חדשה: $\phi \wedge (x \vee y \vee z)$. נשים לב שקיימות בפסוקית החדשה $2^3 = 8$ השמות אפשריות, מתוכן קיימת השמה אחת שהיא לא מספקת: $(F \vee F \vee F)$. כלומר, $\frac{1}{8}$ מתוך ההשמות האפשריות של הפסוקית החדשה הם לא מספקות. מכאן קיימות לפחות $\frac{1}{8}$ השמות לא מספקות, כי אם קיימת השמה מספקת בפסוקית החדשה, עדיין ייתכן כי ב-ϕ ההשמה תהיה לא מספקת ולכן זה "לפחות" $\frac{1}{8}$.

<u>פונקצית הרדוקציה</u>: $f(\phi) = \phi \wedge (x \vee y \vee z)$. כאשר x, y, z הם משתנים חדשים.

<u>תקפות הפונקציה</u>:

- $\phi \in 3 - CNF - SAT \leftarrow$ קיימת השמה מספקת ל-$\phi \leftarrow$ יש השמה מספקת גם ל-$\phi \wedge (x \vee y \vee z) \leftarrow$ נשים לב כי לכל השמה π של ϕ, אם נרחיב אותה ל-$FFF \cdot \pi$ אז נקבל השמה לא מספקת $\leftarrow$ מכיוון של-x, y, z יש שמונה השמות אפשריות, שמינית מכל ההשמות הם לא מספקות $\leftarrow \phi \wedge (x \vee y \vee z) \in L_{1/8}$.
- $\phi \wedge (x \vee y \vee z) \in L_{1/8} \leftarrow$ קיימת השמה מספקת ל-$(x \vee y \vee z) \leftarrow \phi$ יש השמה מספקת ל-$\phi \leftarrow \phi \in 3 - CNF - SAT$.

∎

תרגיל 7 - NAE-K-CNF-SAT - תרגול 7

<u>הגדרה</u>: השפה $NAE - K - CNF - SAT$ (כאשר $NAE = Not\ All\ Equal$) זוהי שפה של פסוקיות $k - CNF$, שיש להן השמה מספקת, אבל בכל פסוקית יש ליטרל שמקבל F.

<u>דוגמה</u>: נתון הפסוק $\phi = (x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.

נגדיר השמות:

1. ההשמה $\pi_1 = (x, y, z) = (T, T, T)$.
2. ההשמה $\pi_2 = (x, y, z) = (T, T, F)$.

נשים לב כי π_1 היא השמה מספקת, אבל לא NAE. מצד שני, ההשמה π_2 היא מספקת וגם NAE.

<u>תרגיל</u>: הוכיח ש: $NAE-3-CNF-SAT \in NPC$.

<u>פתרון</u>: תחילה נוכיח כי $NAE-3-CNF-SAT \in NP$.

נבנה מ"ט א"ד N שתכריע את השפה $NAE-3-CNF-SAT$.

N על קלט $<\phi>$ כאשר ϕ הוא פסוק, תבצע:

1. תנחש השמה π.
2. בדוק השמה π על ϕ:
   a. לכל פסוקית $p \in \phi$ בצע השמה π ובדוק שאכן קיים ליטרל אחד שהוא F - אחרת, <u>דחה</u>.
3. אם π מספקת את ϕ אז <u>קבל</u>. אחרת - <u>דחה</u>.

<u>המכונה ניתנת לחישוב בזמן פולינומי</u>: גודל הניחוש $O(|\pi|)$, בדיקת השמה היא $O(|\phi|)$ ולכן <u>פולינומי</u>.

<u>נכונות הבנייה</u>:

- $<\phi> \in NAE-3-CNF-SAT$ ← קיימת השמה מספקת π כך שלכל פסוקית ב-ϕ קיים ליטרל שהוא F ← קיים ניחוש של השמה π זו ש-N תנחש, תבדוק אותה על ϕ, תבדוק שאכן בכל פסוקית קיים ליטרל שהוא F ← π תצליח בבדיקה של N ותקבל ← $<\phi> \in L(N)$.
- $<\phi> \notin NAE-3-CNF-SAT$ ← לא קיימת השמה מספקת π כך שלכל פסוקית ב-ϕ קיים ליטרל שהוא F ← לכל ניחוש של השמה π ש-N תנחש, היא תבדוק אותה על ϕ:
  - תבדוק שאכן בכל פסוקית קיים ליטרל שהוא F .
  - תבדוק שההשמה π מספקת את ϕ.

  לפחות אחד מהתרחישים לעיל נכשלו בבדיקה של N ולכן היא תדחה ← $<\phi> \notin L(N)$.

כעת נותר להראות כי $NAE-3-CNF-SAT \in NPH$ על ידי שתי רדוקציות:

$$3-CNF-SAT \leq_p NAE-4-CNF-SAT \leq_p NAE-3-CNF-SAT$$

<u>רדוקציה ראשונה</u>: $3-CNF-SAT \leq_p NAE-4-CNF-SAT$.

נגדיר פונקציה כלשהי g שמקבלת פסוקית $\varphi_i = (x \vee y \vee z)$ מ-ϕ ומרחיבה אותה במשנה החדש w.

כלומר: לכל פסוקית φ_i, נגדיר משתנה חדש w בצורה הבאה: $g(x \vee y \vee z) = (x \vee y \vee z \vee w)$.

<u>אבחנה</u>: אם השמה π היא NAE ומספקת אז גם $\bar{\pi}$ היא NAE ומספקת.

<u>פונקצית הרדוקציה</u>: $f(\phi) = \phi'$ כך ש:

- $\phi = \varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_m$
- $\phi' = g(\varphi_1) \wedge g(\varphi_2) \wedge ... \wedge g(\varphi_m)$

<u>הפונקציה ניתנת לחישוב בזמן פולינומי</u>: להוסיף כמות קבועה של משתנה חדש $O(1)$.

<u>הפונקציה תקפה</u>:

- $\phi \in 3-CNF-SAT$ ← קיימת השמה מספקת π ל-ϕ ← ההשמה $\pi \cdot w$ כך ש: $w = F$ היא NAE וגם מספקת ← $\phi' \in NAE-4-CNF-SAT$.
- $\phi' \in NAE-4-CNF-SAT$ ← קיימת השמה מספקת π וגם NAE ← נחלק למקרים:

- אם $\pi(w) = F$, בכל פסוקית יש משתנה "מהמקוריים" שקיבל T ולכן ההשמה מספקת גם את ϕ.
- אם $\pi(w) = T$, אז ניקח את ההופכה שלה $\bar{\pi}$, ולפי האבחנה שהצגנו, גם ההשמה הזאת מספקת את ϕ'

$\leftarrow \phi \in 3 - CNF - SAT$.

<u>רדוקציה שניה</u>: $NAE - 4 - CNF - SAT \leq_p NAE - 3 - CNF - SAT$.

נפעל בדומה להרחבות שעשינו לרדוקציה של $CNF - SAT \leq_p 3 - CNF - SAT$.

נגדיר פונקציה כלשהי g שמקבלת פסוקית $\varphi_i = (x_1 \vee x_2 \vee x_3 \vee x_4)$ מ-ϕ ומפרקת אותה בעזרת המשנה החדש w, לצורה של $3CNF$. כלומר: לכל פסוקית φ_i, נגדיר משתנה חדש w בצורה הבאה:

$$g(x_1 \vee x_2 \vee x_3 \vee x_4) = (x_1 \vee x_2 \vee w_i) \wedge (\bar{w}_i \vee x_3 \vee x_4)$$

<u>פונקצית הרדוקציה</u>: $f(\phi) = \phi'$ כך ש:

- $\phi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_m$
- $\phi' = g(\varphi_1) \wedge g(\varphi_2) \wedge \ldots \wedge g(\varphi_m)$

<u>הפונקציה ניתנת לחישוב בזמן פולינומי</u>: להוסיף כמות קבועה של משתנה חדש $O(1)$ ולפצל לפסוקיות.

<u>הפונקציה תקפה</u>:

- $\phi \in NAE - 4 - CNF - SAT \leftarrow$ קיימת השמה NAE מספקת ל-$\phi \leftarrow$ לכל פסוקית $\varphi_i = (x_1 \vee x_2 \vee x_3 \vee x_4)$ נבדוק:
  - אם $x_1 \neq x_2$ אז נבחר $w_i = x_3$.
  - אם $x_1 = x_2$ אז נבחר $w_i = \overline{x_2}$.

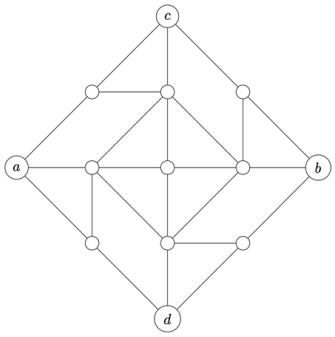
  $\leftarrow$ בכל מקרה, נקבל השמה NAE מספקת ל-$(x_1 \vee x_2 \vee w_i) \wedge (\bar{w}_i \vee x_3 \vee x_4) \leftarrow$ $\phi' \in NAE - 3 - CNF - SAT$.
- $\phi' \in NAE - 3 - CNF - SAT \leftarrow$ קיימת השמה NAE מספקת לכל פסוקית $(x_1 \vee x_2 \vee w_i) \wedge (\bar{w}_i \vee x_3 \vee x_4) \leftarrow$ בה"כ ניתן להניח ש: $w_i = T \leftarrow$ לכן x_1 או x_2 קיבלו $F \leftarrow$ לכן $\bar{w}_i = F \leftarrow$ וגם x_3 או x_4 קיבלו $T \leftarrow$ ההשמה NAE מספקת את $(x_1 \vee x_2 \vee x_3 \vee x_4) \leftarrow$ $\phi \in NAE - 4 - CNF - SAT$.

∎

דוגמה פרקטית להסבר תקיפות כיוון 1:

- אם $\varphi_i = (x_1 \vee x_2 \vee x_3 \vee x_4) = (T \vee T \vee T \vee F)$ אז מאחר ו-$x_1 = x_2$ נבחר $w_i = \bar{x_2} = \bar{T} = F$ ונקבל: $g(x_1 \vee x_2 \vee x_3 \vee x_4) = (T \vee T \vee F) \wedge (T \vee T \vee F)$.
- אם $\varphi_i = (x_1 \vee x_2 \vee x_3 \vee x_4) = (T \vee F \vee T \vee T)$ אז מאחר ו-$x_1 \neq x_2$ נבחר $w_i = x_3$ ונקבל: $g(x_1 \vee x_2 \vee x_3 \vee x_4) = (T \vee F \vee T) \wedge (F \vee T \vee T)$.

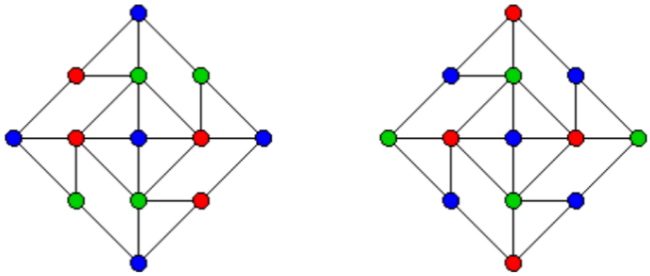
תרגיל 8 - PLANAR COLOR (גרף מישורי צביע) - תרגול 7



לפני שנגיע לרדוקציה הבאה, נתבונן בגרף W משמאל:

הגדרה: **גרף מישורי** הוא גרף שניתן לצייר במישור מבלי שהקשתות תחתוכנה זו את זו (חוץ מאשר בצמתי הגרף).

תכונה 1: בכל 3-צביעה חוקית של W, הצמתים a, b יקבלו את אותו הצבע, וגם הצמתים c, d את אותו הצבע.



תכונה 2: בכל צביעה של צמתים a, b, c, d כך ש: $a = b$ וגם $c = d$, ניתן להרחיב לצביעה יחידה של W כולו.

הגדרת השפה:

$3 - PLANAR - COLOR = \{< G > \mid G\ is\ a\ planar\ graph\ and\ has\ a\ valid\ 3 - coloring\}$

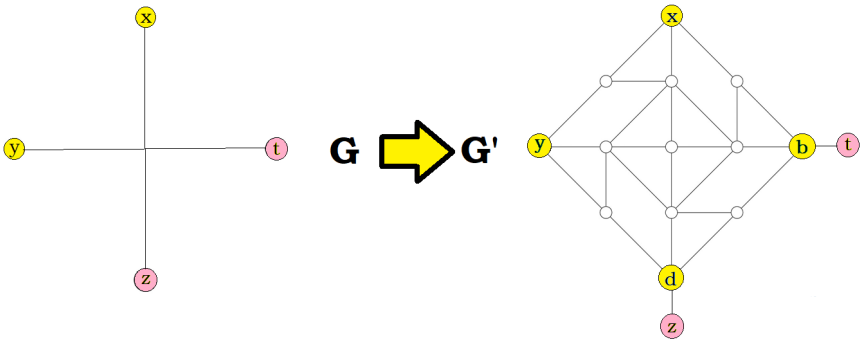
תרגיל: הוכיחו $3 - PLANAR - COLOR \in NPH$.

פתרון: נפתור ברדוקציה: $3 - PLANAR - COLOR \leq_P 3COLOR$.

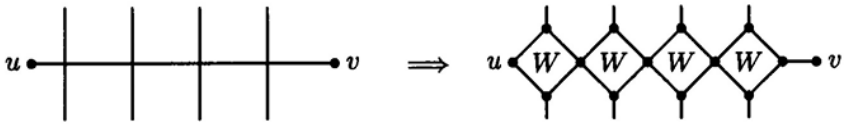
בעצם אנחנו רוצים לקחת גרף ולהפוך אותו למישורי וגם לשמור על תכונת הצביעות.

תיאור הרדוקציה (לא פורמלי):

נצייר את הגרף במישור. בכל מקום שיש חיתוך בין שתי צלעות, נכניס את הגרף W שראינו, כך:



במקרה שיש לנו כמה חיתוכים באותו הצלע אז נבצע כמה העתקות של W:



<u>תקפות הרדוקציה</u>:

- $3COLOR \in < G > \leftarrow$ קיימת 3-צביעה לגרף $G \leftarrow$ הקודקודים המקוריים יישארו עם אותם צבעים $\leftarrow$ בכל עותק של W, נבצע את הקודקודים החיצוניים כך שקודקודים נגדיים יהיו באותו צבע $\leftarrow$ לפי תכונה 2, זה משרה צביעה על כל שאר הקודקודים $\leftarrow$ יש 3-צביעה חוקית ל-G'. $\leftarrow$ $< G' > \in 3 - PLANAR - COLOR$.
- $< G' > \in 3 - PLANAR - COLOR \leftarrow$ יש 3-צביעה חוקית ל-$G' \leftarrow$ לפי תכונה 1, בכל עותק של W הקודקודים הנגדיים קיבלו את אותו הצבע $\leftarrow$ כל זוג קודקודים שהיה מחובר בצלע בגרף המקורי, קיבל צבע שונה כל אחד $\leftarrow$ הצביעה על הקודקודים המקוריים היא צביעה חוקית $\leftarrow$ $< G > \in 3COLOR$.

<u>טריק חשוב</u>: "משפט ארבעת הצבעים" מבטיח שלכל גרף מישורי יש 4-צביעה.
כלומר, השפה $4 - PLANAR - COLOR \in P$ כי בעזרת המשפט, אפשר לבנות מ"ט דטרמיניסטית שבהינתן גרף G, מספיק לה לבדוק כי G מישורי, אם כן, אז בהכרח קיים 4-צביעה ל-G.

תרגיל 9 - kCOL - תרגול 7

<u>הגדרה</u>: נגדיר את משפחת השפות: $kCOL = \{< G > \mid G\ is\ a\ graph\ and\ has\ a\ k - coloring\}$
<u>תרגיל 1</u>: האם $1COL \in P, NP, NPC$?
<u>פתרון 1</u>: $1COL \in P$. ניתן לצבוע גרף בצבע בודד רק אם לא קיימות בו צלעות בכלל.

<u>תרגיל 2</u>: האם $2COL \in P, NP, NPC$?
<u>פתרון 2</u>: $2COL \in P$. נראה אלגוריתם באופן לא פורמלי: נבנה מ"ט דטרמיניסטית M המכריעה את $2COL$.
M על קלט $< G >$ תבצע:

1. נבחר קודקוד u באקראי ונבצע אותו בצבע "1".
2. מרגע זה, ניתן להכריח לצבוע את הקודקודים השכנים ל-u בצבע "2", לאחר מכן לכל אחד מהשכנים נצבע גם את השכנים שלהם ב-"1" וככה הלאה...
3. נמשיך לצבוע עד שנגיע לסתירה או שנסיים את כל רכיב הקשירות.
4. נבצע עבור כל רכיבי הקשירות ב-G.
5. אם צבענו את כל G מבלי להגיע לסתירה - <u>קבל</u>. אחרת - <u>דחה</u>.

<u>הערה חשובה</u>: אם ביקשו ממנו גרף k-צביע כאשר בגרף יש רק n קודקודים כך ש: $k \geq n$ אז קיימים יותר צבעים מקודקודים ולכן אפשר לפתור באופן דטרמיניסטי ובזמן פולינומי מקרים פרטיים כאלה.

בעיות חיפוש למול בעיות הכרעה

שפות ב-NP כבעיות הכרעה

כל השפות שראינו ב-NP עד כה היו מהצורה: $\{< Object > \mid \exists(a\ certain\ property\ \pi)\ in\ Object\}$.

- ב-$CLIQUE$ היינו צריכים להכריע האם קיימת קליקה בגודל הנתון.
- ב-VC שאלנו האם קיים כיסוי קודקודים בגודל הנדרש.
- ב-SAT שאלנו האם קיימת השמה מספקת.

כל השפות/בעיות הללו הן **בעיות הכרעה**.

הגדרה - בעיות חיפוש

לכל אחת מבעיות ההכרעה שהצגנו, ניתן להתאים בעיית חיפוש:

- **<u>קלט</u>**: $< Object >$.
- **<u>פלט</u>**: אם π קיים - נחזיר אותו. אחרת - נחזיר סימן מוסכם (למשל את המחרוזת "לא קיים π כנדרש").

בעיות אלו הן **בעיות חיפוש** והן שייכות למחלקה FNP המכילה יחסים ולא שפות.

בעיית החיפוש המתאימה ל-CNF-SAT

<u>הגדרה</u>: בעיית החיפוש המתאימה ל-$CNF = SAT$ מוגדרת כך:

- **<u>קלט</u>**: פסוק הנתון בצורת CNF.
- **<u>פלט</u>**: השמה המספקת את הפסוק (אם קיימת).

<u>משפט</u>: אם $P = NP$ אזי קיים אלגוריתם <u>דטרמיניסטי</u> פולינומי לבעיית החיפוש של $CNF - SAT$.

(כלומר: אם $CNF - SAT \in P$ אז קיים אלגוריתם **דטרמיניסטי ופולינומי** לבעיית החיפוש המתאימה ל-$CNF - SAT$).

<u>הוכחת המשפט</u>: ראשית, בהינתן פסוק ϕ בצורת CNF עם n משתנים $(x_1,...,x_n)$, ניתן לבחור משתנה מסוים ולהציב עבורו ערך אמת מסוים. ונקבל פסוק CNF חדש עם $n - 1$ משתנים. באופן לא מדויק, נכתוב את ערך האמת בתוך הפסוק. נשתמש בכללי הלוגיקה הבאים: (**השלימו**)

$$a \vee 0 \equiv a \quad a \wedge 0 \equiv 0$$
$$a \vee 1 \equiv 1 \quad a \wedge 1 \equiv a$$

נגדיר:

- ϕ_0 הוא הפסוק המתקבל מ-ϕ ע"י הצבת ערך 0 במשתנה עם האינדקס הקטן ביותר.
- ϕ_1 הוא הפסוק המתקבל מ-ϕ ע"י הצבת ערך 1 במשתנה עם האינדקס הקטן ביותר.

<u>הוכחה</u>: ננית ש: $P = NP$. אזי קיים אלגוריתם דטרמיניסטי פולינומי M המכריע את $CNF - SAT$.
נתאר אלגוריתם $SearchSAT$ דטרמיניסטי המחזיר השמה מספקת לפסוק CNF נתון, או מחזיר את המחרוזת "לא קיימת השמה מספקת". נגדיר השמה לפסוק כמילה מעל $\{0,1\}$, כך שהמקום ה-i במילה יהיה ערך האמת של המשתנה ה-i.

<u>האלגוריתם</u> $SearchSAT$ על קלט ϕ כאשר ϕ הוא פסוק CNF עם n משתנים, תבצע:

1. סמלץ את M על הקלט ϕ.
   a. אם $M(\phi) = 0$ אז החזר "לא קיימת השמה מספקת".
   b. אחרת, המשך לשלב 2.
2. אתחל השמה ריקה: $\pi = \varepsilon$.
3. כל עוד $n > 0$ תבצע:
   a. הצב ערך '0' במשתנה בעל האינדקס הנמוך ביותר של ϕ ונקבל את הפסוק ϕ_0.
   b. סמלץ את ריצת M על הקלט ϕ_0:
      i. אם $M(\phi_0) = 1$ אז בצע:
         1. $\pi = 0 \cdot \pi$
         2. $\phi = \phi_0$
      ii. אחרת, אם $M(\phi_0) = 0$ אז בצע:
         1. $\pi = 1 \cdot \pi$
         2. $\phi = \phi_0$
   c. $n = n - 1$.
4. החזר את ההשמה π.

<u>הוכחת נכונות האלגוריתם</u>:

- אם $\phi \in CNF - SAT$ ← בכל שלב, המכונה M יכריע עבור משתנה יחיד האם הוא צריך לקבל ערך של 0 או 1 ← בסוף התהליך תהיה לנו השמה מספקת.
- אם $\phi \notin CNF - SAT$ ← בשלב 1 המכונה M תחזיר 0 ← $SearchSAT$ יחזיר "לא קיימת השמה".

<u>סיבוכיות זמן ריצה</u>: נסמן ב-$t(\cdot)$ את הסיבוכיות של הריצה של M, על פי ההנחה, $t(\cdot)$ הוא פולינום.
האלגוריתם $SearchSAT$ קורא ל-M כ-n פעמים, לכן הסיבוכיות היא $O(n \cdot t(n))$ - <u>פולינומית</u>.
נשים לב שבכל שלב חישוב האלגוריתם "יורד" רמה אחת בעץ הבינארי שציירנו, ו-M מכוון אותו בכל צומת לאיזה בן לרדת. עומק העץ הוא n כך שהסיבוכיות של האלגוריתם שלנו היא אכן $O(n \cdot t(n))$.

דוגמאות לבעיות חיפוש - אלגוריתם דטרמיניסטי ל-CLIQUE

(שאלת מבחן): נניח שחוקר מוכשר הצליח למצוא אלגוריתם דטרמיניסטי פולינומי ל-$CLIQUE$, ומימש אותו באמצעות מכונת טיורינג M_{CLIQUE}. תארו (במילים) מימוש של מ"ט יעילה שבהינתן קלט $< G, k >$, מחזירה קליק ב-G שגודלו k, או מחזירה "לא קיים", אם לא קיים קליק כזה.

<u>פתרון</u>:

<u>הרעיון</u>: נוריד קודקודים וכל פעם שהאלגוריתם M_{CLIQUE} יחזיר $FALSE$, זה היה קודקוד נחוץ ולכן נחזיר אותו.

- בדיקה ראשונית, אם הקלט $< G, k >$ שייך לשפה $CLIQUE$.
- בכל שלב נוריד קודקוד, ואת כל הצלעות החלות בו, יהי G' הגרף המתקבל לאחר ההורדה.
- נבדוק אם $< G', k >$ בשפה $CLIQUE$, אם כן, נמשיך, אחרת נחזור אל G.
- נעצור כאשר גודל הגרף G' הוא בדיוק k קודקודים.
- נחזיר את קבוצת הקודקודים של G'.

<u>האלגוריתם</u> $SearchCLIQUE$ על קלט $< G, k >$ כאשר G הוא גרף ו-k טבעי, תבצע:

1. סמלץ את M_{CLIQUE} על קלט $< G, k >$.
   a. אם $M_{CLIQUE}(< G, k >) = 0$, החזר "לא קיים".
   b. אחרת, עבור לשלב 2.
2. הגדר קבוצת קודקודים ריקה $S = \{\}$.
3. עבור $1 \leq n \leq |V(G)|$ בצע:
   a. הוצא קודקוד v_n והגדר $G_n = G_{n-1} \backslash \{v_n\}$.
   b. סמלץ את ריצת M_{CLIQUE} על $< G_n, k >$.
      i. אם $M_{CLIQUE}(< G_n, k >) = 0$ אז:
         1. $S = S \cup \{v_n\}$.
         2. $G_{n-1} = G_n \cup \{v_n\}$.
         c. $n = n + 1$.
4. החזר את ההשמה S.

<u>הוכחת נכונות האלגוריתם</u>:

- אם $< G, k > \in CLIQUE$ ← קיימת קליקה בגודל k ← היות והאלגוריתם בודק שייכות של כל קודקוד בנפרד, הוא ימצא את הקליקה.
- אם $< G, k > \notin CLIQUE$ ← האלגוריתם יחזיר "לא קיים" כבר בשורה הראשונה.

<u>סיבוכיות זמן הריצה</u>: נניח כי M_{CLIQUE} רץ בזמן t. אזי האלגוריתם רץ בזמן $O(nt)$ - <u>פולינומי</u>.

דוגמאות לבעיות חיפוש - אלגוריתם דטרמיניסטי ל-HAMPATH

(שאלת מבחן): נניח כי קיים אלגוריתם דטרמיניסטי פולינומי (הממומש במ"ט $M_{HAMPATH}$) המכריע את השפה:

$$HAMPATH = \{< G > \mid G\ is\ a\ graph\ that\ has\ a\ Hamiltonian\ path\}$$

<u>סעיף א'</u>: מה ההשלכה של קיום האלגוריתם הזה לשאלה האם $P = NP$?

<u>סעיף ב'</u>: הראו אלגוריתם דטרמיניסטי פולינומי אשר בהינתן קלט $< G >$ מחזיר פרמוטציה של קודקודי הגרף אשר מהווה מסלול המילטוני, אם קיים כזה, אחרת - מחזירה את המחרוזת "לא קיים מסלול המילטוני".
הסבירו את פעולת האלגוריתם במילים שלכם. אין צורך להוכיח נכונות, אך יש להראות שהסיבוכיות הינה פולינומית.

<u>פתרון א'</u>: היות ו-$HAMPATH \in NPC$, קיום אלגוריתם דטרמיניסטי פולינומי מעיד כי $HAMPATH \in P$ וגם כל שאר השפות ב-NP מרדוקציה. לכן, $P = NP$.

<u>פתרון ב'</u>:

<u>הרעיון</u>:

- בדיקה ראשונית, אם הקלט $< G >$ בשפה $HAMPATH$.
- בכל שלב נוריד צלע ונקבל את G'.
- נבדוק אם $< G' >$ בשפה, אם כן, נמשיך, אחרת נחזור אל G.
- נעצור כאשר כמות הצלעות בגרף G' היא בדיוק $n - 1$.
- נשתמש בקבוצת הצלעות שנותרה כדי להחזיר את הסידור של הקודקודים על המסלול ההמילטוני בגרף.

<u>אלגוריתם</u> $SearchHAMPATH$ על קלט $< G >$ כאשר G הוא גרף, תבצע:

1. סמלץ את $M_{HAMPATH}$ על קלט $< G >$:
   a. אם $M_{HAMPATH}(< G >) = 0$ - החזר "לא קיים מסלול המילטוני".
   b. אחרת, המשך לשלב 2.
2. נגדיר קבוצת צלעות ריקה $S = \{\}$.
3. עבור $1 \leq m \leq |E(G)|$ בצע:
   a. הוצא צלע e_m והגדר $G_m = G_{m-1} \backslash \{e_m\}$.
   b. סמלץ את ריצת $M_{HAMPATH}$ על קלט $< G_m >$.
      i. אם $M_{HAMPATH}(< G_m >) = 0$ אז:
         1. $S = S \cup \{e_m\}$.
         2. $G_{m-1} = G_m \cup \{e_m\}$.
      c. $m = m + 1$.
4. החזר את S.

אז אחרי כמעט 4 שנים של לימודים אינטנסיביים של 12 שעות למידה ביום, גם אני מסיים את התואר. זה הקורס האחרון שלי. חשוב לי להגיד שאין דבר העומד בפני הרצון. התחלתי את התואר בזלזול רב וחוסר רצון ללמוד, בנוסף יש לי הרבה קשיי למידה ולוקח לי זמן להבין דברים. עם הזמן תפסתי את עצמי ברצינות והתחלתי לקחת את ענייני הסיכומים ברצינות כדי להוכיח לעצמי שאני יכול הכל. אם יש לכם מטרה שאולי מרגישה לכם בהתחלה מאוד מטופשת כמו האופן שבו אני סיכמתי כל קורס, אני ממליץ לכם דווקא כן ללכת עליה ולהלחם עליה כי בסוף הדרך זה נותן את הסיפוק הגדול ביותר (וגם אחלה של תוסף לרזומה).
בהצלחה לכולנו בהמשך, זה רק ההתחלה! 😀

וכן, נותר רק לכתוב בפעם האחרונה: **חלאס.**