תוכן העניינים

1 מכונות טיורינג

הגדרה 1: מכונת טיורינג

מכונת טיורינג (מ"ט) היא שביעיה $M=(Q,\Sigma,\Gamma,\delta,q_0,\text{acc},\text{rej})$.

Q	קבוצת מצבים סופיות	
Σ	א"ב קלט סופי	$\_\notin\Sigma$
Γ	א"ב סרט סופי	$\Sigma\subseteq\Gamma\,,\ \_\in\Gamma$
δ	פונקציית המעברים	$\delta:(Q\backslash\{\text{rej},\text{acc}\}\times\Gamma\to Q\times\Gamma\times\{L,R\}$
q_0	מצב התחלתי	
acc	מצב מקבל	
rej	מצב דוחה	

הגדרה 2: קונפיגורציה

תהי $M=(Q,\Sigma,\Gamma,\delta,q_0,\text{acc},\text{rej})$ מכונת טיורינג.

קונפיגורציה של M הינה מחרוזת

$$uq\sigma \mathrm{v}\ , \quad u,\mathrm{v}\in\Gamma^*,\ \sigma\in\Gamma,\ q\in Q\ .$$

משמעות:

q מצב המכונה,
σ הסימון במיקום הראש
u תוכן הסרט משמאל לראש,
v תוכן הסרט מימין לראש.

הגדרה 3: גרירה

תהי $M=(Q,\Sigma,\Gamma,\delta,q_0,\mathrm{acc},\mathrm{rej})$ מכונת טיורינג, ותהיינה c_1 ו- c_2 קונפיגורציות של M.
נסמן

$$c_1\vdash_M c_2$$

(במילים, c_1 גורר את c_2) אם כשנמצאים ב- c_1 עוברים ל- c_2 בצעד בודד.

נסמן

$$c_1\vdash_M^* c_2$$

אם ניתן לעבור מ- c_1 ל- c_2 ב- 0 או יותר צעדים.

הגדרה 4: קבלה ודחייה של מחרוזת

תהי $M=(Q,\Sigma,\Gamma,\delta,q_0,\mathrm{acc}\,,\ \mathrm{rej})$ מכונת טיורינג, ו- $w\in\Sigma^*$ מחרוזת.
נאמר כי:

- M **מקבלת** את w אם $\quad q_0w\vdash_M^* u\ \mathrm{acc}\,\sigma\ \mathrm{v}$
- M **דוחה** את w אם $\quad q_0w\vdash_M^* u\ \mathrm{rej}\ \sigma\ \mathrm{v}$

כאשר $\Gamma\ni\sigma, \Gamma^*\ni u, \mathrm{v}$ כלשהם.

הגדרה 5: הכרעה של שפה

תהי $M=(Q,\Sigma,\Gamma,\delta,q_0,\mathrm{acc}\,,\mathrm{rej})$ מכונת טיורינג, ו- $L\subseteq\Sigma^*$ שפה.
נאמר כי M **מכריעה** את L אם לכל $w\in\Sigma^*$ מתקיים

- $L\ni w \Leftarrow M$ מקבלת את w.
- $L\not\ni w \Leftarrow M$ דוחה את w.

הגדרה 6: קבלה של שפה

תהי $M=(Q,\Sigma,\Gamma,\delta,q_0,\mathrm{acc}\,,\ \mathrm{rej})$ מכונת טיורינג, ו- $L\subseteq\Sigma^*$ שפה.
נאמר כי M **מקבלת** את L אם לכל $w\in\Sigma^*$ מתקיים

- אם $L \in w$ אז M מקבלת את w.
- אם $L \notin w$ אז M לא מקבלת את w.

במקרה כזה נכתוב ש- $L(M) = L$.

הגדרה 7: חישוב פונקציות

תהי $M = (Q, \Sigma, \Gamma, \delta, q_0, \text{acc}, \text{rej})$ מכונת טיורינג ותהי $f : \Sigma_1^* \to \Sigma_2^*$.
נאמר כי M **מחשבת** את f אם:

- $\Sigma = \Sigma_1$,$\Sigma_2 \subset \Gamma$,
- לכל $w \in \Sigma_1^*$ מתקיים $q_0 w \vdash_M^* \text{acc} f(w)$.

2 וריאציות של מכונות טיורינג

הגדרה 8: מודל חישוב

מודל חישובי = אוסף של מכונות שעבורם מוגדרים המושגים של הכרעה וקבלה של שפות.

הגדרה 9: מודלים שקולים חישובית

יהיו A, B מודלים חישוביים. נאמר כי A ו- B שקולים אם לכל שפה L:

- קיימת מכונה במודל A שמכריעה את L אם"ם קיימת מכונה כזו במודל B.
- קיימת מכונה במודל A שמקבלת את L אם"ם קיימת מכונה כזו במודל B.

הגדרה 10: מכונות שקולות חישובית

שתי מכונות הן שקולות חישובית אם הן מקבלות ודוחות בדיוק את אותן המילים.

משפט 1: מכונת טיורינג עם סרט ימינה בלבד

מודל מ"ט טם סרט אינסופי לכיוון אחד בלבד (מודל O) שקול למודל אינסופי בשני הכיוונים (מודל T).
כלומר, לכל שפה L:

- יש מ"ט ממודל O שמקבלת את L אם"ם יש מ"ט במודל T שמקבלת את L.
- יש מ"ט ממודל O שמכריעה את L אם"ם יש מ"ט במודל T שמכריעה את L.

משפט 2: מכונת טיורינג מרובת סרטים

במכונת טיורינג מרובת סרטים:

- יתכנו מספר סטרים.

מספר הסרטים סופי וקבוע מראש בזמן בניית המ"ט, ואינו תלוי בקלט או במהלך החישוב.

- לכל סרט יש ראש נפרד.
- הפעילות (תנועה וכתיבה) בכל סרט נעשית בנפרד.

  בפרט, הראשים יכולים לזוז בכיוונים שונים בסרטים שונים.
- ישנו בקר מרכזי יחיד, שקובע את הפעילות בכל אחד מהסרטים, על סמך המידע שמתקבל מכל הסרטים.

  לכן, תוכן סרט אחד יכול להשפיע על הפעילות בשאר הסרטים.
- בתחילת החישוב, הקלט נמצא בסרט הראשון ושאר הסרטים ריקים.

משפט 3:

לכל k, המודל של מ"ט עם k סטרים שקול חישובי למודל של מ"ט עם סרט אחד.

משפט 4:

קבלה ודחייה של מחרוזות:
עבור מ"ט לא דטרמיניסטית N ומחרוזת w:

- N **מקבלת** את w אם קיים חישוב של N על w שמגיע למצב מקבל.
- N **דוחה** את w אם כל החישובים של N על w עוצרים במצב דוחה.

הכרעה וקבלה של שפות:
עבור מ"ט לא דטרמיניסטית N ושפה L:

- N **מכריעה** את L אם N מקבלת אץ כל המילים ב- L ודוחה את כל המילים שאינן ב- L.
- N **מקבלת** את L אם N מקבלת אץ כל המילים ב- L ולא מקבלת את כל המילים שאינן ב- L.

משפט 5:

לכל מ"ט לא דטרמיניסטית קיימת מ"ט דטרמיניסטית שקולה.

3 התזה של צ'רץ'-טיורינג

שמות נרדפים לשפות כריעות ושפות קבילות

שפות כריעות	Decideable languages	שפות קבילות	Acceptable languages
שפות רקורסיביות	Recursive languages	שפות ניתנות לזיהוי	recognizable languages
		שפות כריעות למחצה	Semi-deidable languages
			Partially-deidable languages
		שפות הניתנות למנייה רקורסיביות	Recursively enumerable languages

משפט 6: סגירות שפות כריעות

השפות הכריעות סגורות תחת:

- איחוד
- חיתוך
- משלים
- שרשור
- סגור קלין

משפט 7: סגירות שפות קבילות

- איחוד
- חיתוך
- שרשור
- סגור קלין

משפט 8: היחס בין הכרעה לקבלה

אם שפה הינה כריעה אז היא קבילה.
אם שפה והמשלים שלה קבילות אז היא כריעה.

הגדרה 11: שפת סימפל

<u>משתנים</u>

- טבעיים: `i,j,k,...`
  מקבלים כערך מספר טבעי.
- מערכים: `A[],B[],C[],...` בכל תא ערך מתוך א"ב Γ אין סופיים.
- אתחול: הקלט נמצא בתאים הראשונים של `A[]`.
  כל שאר המשתנים מאותחלים ל- `0`.

<u>פעולות</u>

- השמה בקבוע:
  `i=3, B[i]="#"`

- השמה בין משתנים:

  `i=k, A[k]=B[i]`

- פעולות חשבון:

  `x = y + z , x = y - z , x = y.z`

<u>תנאים</u>

- `B[i]==A[j]`

  (מערכים).

- `x >= y`

  (משתנים טבעיים).

כל משתנה מופיע רק פעם אחת בכל פעולה או תנאי.

<u>זרימה</u>

- סדרה פקודות ממוספרות.
- `goto`: מותנה ולא מותנה.
- `stop` עצירה עם ערך חזרה.

```
1 one = 1
2 zero = 0
3 B[zero] = "0"
4 i=0
5 j=i
6 if A[i] == B[zero] goto 9
7 i=j + one
8 goto 3
9 C[one] = A[j]
10 if C[one] ==  A[zero] goto 12
11 stop(0)
12 stop(1)
```

הגדרה 12: קבלה ודחייה של מחרוזת בשפה SIMPLE

עבור קלט
w ותוכנית
P בשפת SIMPLE. נאמר כי

- P **מקבלת** את w אם הריצה של P על w עוצרת עם ערך חזרה 1.
- P **דוחה** את w אם הריצה של P על w עוצרת עם ערך חזרה 0.

הגדרה 13: הכרעה וקבלה של שפות

עבור שפה L ותוכנית P בשפת SIMPLE. נאמר כי

- P **מכריעה** את L אם היא מקבלת את המילים שב- L ודוחה את אלה שלא ב- L.
- P **מקבלת** את L אם היא מקבלת את כל ורק המילים ב- L.

משפט 9:

המודלים של מכונת טיורינג ותוכניות SIMPLE שקולים.

משפט 10: מ"ט ותוכניות מחשב

מ"ט חזקה לפחות כמו תוכנית משחב.
כל תוכנית מחשב ניתנת למימוש במ"ט.
לכן, כל שפה שהינה כריעה ע"י מחשב היא כם כריעה ע"י מ"ט.
וכמו כן, שפה שהינה קבילה ע"י מחשב היא גם קבילה ע"י מ"ט.

הגדרה 14: דקדוקים כלליים

בדקדוק כללי, בצד שמאל של כלל יצירה יכולה להופיעה מחרוזת (לא ריקה) כלשהי.
פורמלית, כלל יצירה בדקדוק כללי הוא מהצורה

$$\gamma \to u$$

כאשר $\gamma \in (V \cup \Sigma)^+$, $u \in (V \cup \Sigma)^*$.

משפט 11:

תהי L שפה. L קבילה אם"ם קיים דקדוק כללי G כך ש- $L(G) = L$.

משפחת שפות	דקדוק	מודל חישובי
קבילות	כללי	מכונת טיורינג
חסרות הקשר	חסר הקשר	אוטומט מחסנית
רגולריות	רגולרי	אוטומט סופי

משפט 12:

כל שפה חסרת הקשר הינה כריעה.

משפט 13: התזה של צ'רץ' טיורינג

התזה של צ'רץ' טיורינג מודל מ"ט מגלם את המושג האבסטרקרטי של "אלגוריתם".
כלומר, כל אלגוריתם שניתן לתיאור כתהליך מכניסטי שבו:

- התהליך מתבצע כסדרה של צעדים.
- כל צעד מצריך כמות סופית של "עבודה".

ניתן גם לתיאור כמ"ט.
בפרט, אין מודל מכניסטי / אוטומטי יותר ממ"ט.

4 אי-כריעות

הגדרה 15: השפה ATM

$$ATM = \{(P, w) \mid P(w) = 1\} \ .$$

השפה ATM כוללת את כל הזוגות של מחרוזות P, w כך ש:

- P היא קוד (תקין) של תוכנית.
- w מחרוזת.
- מתקיים שאם מריצים את התוכנית P על הקלט w אז התוכנית עוצרת עם ערך חזרה 1.

הגדרה חלופית:

$$A_{TM} = \{\langle M, w\rangle \mid \text{מכונת טיורינג שמקבלת את } w \ M\}$$

השפה A_{TM} כוללת את כל הזוגות של מחרוזות $\langle M, w\rangle$ של כל מכונת טיורינג M וכל קלט w כך ש- M מקבלת את w.

סיכום 1: התוכנה U

התוכנה U היא תוכנה שמקבלת כקלט זוג מחרוזות P, w ופועלת כך:

- U מחזירה את ערך החזרה שהתקבל מהריצה של P על w.
- מריצה את התוכנה P על הקלט w (במקרה שבו P אינה תוכנית מחשב תקינה אז U מחזירה ערך 0).

נשים לב שאם P לא עוצרת על w אז גם U לא עוצרת על הזוג P, w.
התוכנה U פועלת באופן דומה לאופן שבה מערכת ההפעלה מפעילה תוכנות אחרות.

התוכנה U נקראת גם "תוכנה אוניברסלית" (או, בעולם מ"ט "מ"ט אוניברסלית") כיוון שהיא תוכנה אחת שמדמה כל תוכנה אחרת.

U היא תוכנית שמקבלת את ATM.

כלומר:

$$L(U) = ATM \ .$$

מסקנה: ATM קבילה.

שיטת ההוכחה:
אם שפה הינה קבילה אבל לא כריעה, אז המשלים שלה בהכרח אינה קבילה.
לכן, בשביל להוכיח ששפה אינה קבילה, די להוכיח שהשפה לא כריעה, והמשלים שלה כן קבילה.

הגדרה 16: השפה $HALT$

$$HALT = \big\{(P,w) \ \big| \ P(w)\downarrow\big\} \ .$$

השפה HALT כוללת את כל הזוגות של מחרוזות P,w כך ש:

- P היא קוד (תקין) של תוכנית.
- w מחרוזת.
- מתקיים שאם מריצים את התוכנית P על הקלט w אז התוכנית עוצרת (הסימון $\downarrow$ מסמן עצירה).)

בעיית העצירה קבילה אבל לא כריעה.
כיוון שכך, המשלים שלה לא כריעה ולא קבילה.

הגדרה חלופית:

$$HALT_{TM} = \big\{\langle M,w\rangle \ \big| \ \text{מכונת טיורינג שעוצרת על } w \ M\big\}$$

השפה $HALT_{TM}$ כוללת את כל הזוגות של מחרוזות $\langle M,w\rangle$ של כל מכונת טיורינג M וכל קלט w כך ש-
M עוצרת על w.

הגדרה 17: השפה E

$$E = \big\{P \ \big| \ L(P) = \emptyset\big\}$$

השפה E כוללת את כל המחרוזות P כך ש-

- P היא קוד (תקין) של תוכנית.
- השפה של P ריקה.

כלומר, לכל קלט w, הריצה של P על w לא מחזיריה 1.

הגדרה חלופית:

$$E_{TM} = \big\{\langle M\rangle \ \big| \ L(M) = \emptyset \ \text{מכונת טיורינג שעומדת בתנאי} \ M\big\}$$

השפה E_{TM} כוללת את כל מחרוזות $\langle M\rangle$ של כל מכונת טיורינג M כך ש- M לא מקבלת אף מילה.
במילים אחרות, השפה של M ריקה: $L(M) = \emptyset$.

הגדרה 18: השפה EQ_{TM}

$$EQ = \{(P_1, P_2) \mid L(P_1) = L(P_2)\} \ .$$

השפה EQ כוללת את כל זוגות המחרוזות P_1, P_2 כך ש:

- P_1, P_2 הינן קודים (תרינים) של תוכניות.
- השפות של P_1, P_2 זהות.

כלומר, P_1, P_2 מקבלות בדיוק את אותן המילים.

הגדרה חלופית:

$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2) \text{ מכונות טיורינג } M_1, M_2\}$$

השפה EQ_{TM} כוללת את כל זוגות של מכונות טיורינג $\langle M_1, M_2\rangle$ שמקבלות בדיוק אותן המילים.
במילים אחרות, השפות של M_1 ו- M_2 זהות: $L(M_1) = L(M_2)$.

	כריעה	קבילה
ATM	$\times$	$\checkmark$
$\overline{ATM}$	$\times$	$\times$
$HALT$	$\times$	$\checkmark$
$\overline{HALT}$	$\times$	$\times$
E	$\times$	$\times$
$\overline{E}$	$\times$	$\checkmark$
EQ	$\times$	$\times$
$\overline{EQ}$	$\times$	$\times$

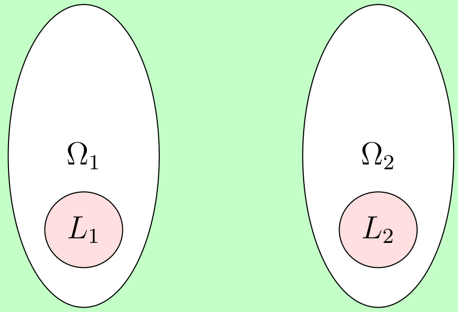
הגדרה 19: הרדוקציה

רדוקציית התאמה (many to one reduction)
מקבוצה $L_1 \subseteq \Omega_1$ לקבוצה $L_2 \subseteq \Omega_2$
הינה **פונקציה**

$$R : \Omega_1 \to \Omega_2$$

כך שלכל $x \in \Omega_1$ מתקיים:

$$x \in L_1 \quad \Leftrightarrow \quad R(x) \in L_2 \ .$$



סימון: $L_1 \le_m L_2$ ריימת רדוקציה התאמה ניתנת לחישוב מ- L_1 ל- L_2.

משפט 14: משפט הרדוקציה

טענה:
אם:

- L_2 כריעה
- $L_1 \le L_2$

אז L_1 כריעה.

מסקנה:
אם:

- L_1 לא כריעה
- $L_1 \le L_2$

אז L_2 לא כריעה.

מתכון להוכחה ששפה L_2 לא כריעה:

1. בחר שפה L_1 לא כריעה.
2. מצא רדוקציית התאמה ניתנת לחישוב מ- L_1 ל- L_2.

טענה:
אם:

- L_2 קבילה
- $L_1 \le L_2$

אז L_1 קבילה.

מסקנה:
אם:

- L_1 לא קבילה
- $L_1 \le L_2$

אז L_2 לא קבילה.

מתכון להוכחה ששפה L_2 לא קבילה:

1. בחר שפה L_1 לא קבילה.
2. מצא רדוקציית התאמה ניתנת לחישוב מ- L_1 ל- L_2.

משפט 15: תכונות של רדוקציות

A	$\le_m$	B
כריעה	$\Leftarrow$	כריעה
לא כריעה	$\Rightarrow$	לא כריעה

A	$\le_m$	B
קבילה	$\Leftarrow$	קבילה
לא קבילה	$\Rightarrow$	לא קבילה

משפט 16: לכל שפה קיימת רדוקציה ל- A_{TM}

מכל שפה כריעה A קיימת רדוקציה חישובית ל- A_{TM}.

כלומר

$$A \le_m A_{TM} \ .$$

משפט 17: רדוקציה משפות כריעות

מכל שפה כריעה קיימת רדוקציה חשיבה לכל שפה אחרת שאינה $\emptyset$ או Σ^*.

הגדרה 20:

$$NOTREG = \left\{ P \;\middle|\; \text{.לא רגולרית } L(P) \right\} \ .$$

השפה NOT-REG כוללת את כל המחרוזות P כך ש:

- P הינה קוד (תקין) של תוכנית.
- השפה של P לא רגולרית.

הגדרה חלופית:

$$NOTREG_{TM} = \left\{ \langle M \rangle \;\middle|\; \text{.לא רגולרית } L(M) \right\} \ .$$

השפה $NOTREG_{TM}$ כוללת את כל המחרוזות $\langle M \rangle$ של מ"ט M כך שהפשה של M לא רגולרית.

משפט 18: השפה $NOT-REG$ אינה קבילה.

השפה $NOT-REG$ אינה קבילה.

5 סיבוכיות זמן

הגדרה 21: זמן הריצה

זמן הריצה של מכונת טיורינג M על קלט w הוא מספר צעדי החישוב ש- M מבצעת על w.

הגדרה 22: סיבוכיות זמן ריצה

תהי M מ"ט דטרמיניסטית אשר עוצרת על כל קלט. **הזמן הריצה** או **הסיבוכיות זמן** של M היא פונקציה $f:\mathbb{N}\to\mathbb{N}$, כאשר $f(n)$ המספר צעדי חישוב המקסימלי ש- M מבצעת על קלט w של אורך n.

אם $f(n)$ זמן הריצה של M, אומרים כי M רץ בזמן $f(n)$ וש- M היא $f(n)$ זמן מכונת טיורינג

הגדרה 23: סימון אסימפטוטי

תהיינה f,g פונקציות

$$f:\mathbb{N}\to\mathbb{R}^+ \ , \qquad g:\mathbb{N}\to\mathbb{R}^+$$

כאשר $\mathbb{R}^+$ הממשיים הלא שליליים.

אומרים כי

$$f(n)=O\left(g(n)\right)$$

אם קיימים שלמים c ו- n_0 עבורם לכל $n\geq n_0$ מתקיים

$$f(n)\leq cg(n)\ .$$

אם $f(n)=O\left(g(n)\right)$ אורמים כי $g(n)$ חסם עליון אסימפטוטי של $f(n)$.

משפט 19:

לכל $a,b,n\in\mathbb{R}$,

$$\log_a n=\frac{\log_b n}{\log_b a}\ .$$

ז"א מעבר מבסיס a לבסיס b משנה את הערך של הלוגריתם עד פקטור של $\dfrac{1}{\log_b a}$. מכיוון ששינוי של המקדם לא משנה את החסם עליון אסימפטוטי, במידה שההתנהגות האסימפטוטית של פונקציה כלשהי היא $\log_a n$ אנחנו פשוט רושמים $O(\log n)$ ללא הבסיס.

הגדרה 24:

תהיינה f,g פונקציות

$$f:\mathbb{N}\to\mathbb{R}^+ \ , \qquad g:\mathbb{N}\to\mathbb{R}^+\ .$$

אומרים כי

$$f(n)=o\left(g(n)\right)$$

אם

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=0\ .$$

במילים פשוטות, $f(n)=o\left(g(n)\right)$ אם לכל מספר ממשי $c>0$ קיים מספר ממשי n_0 כך ש- $f(n)<cg(n)$ לכל $n\geq n_0$.

הגדרה 25: מחלקה של סיבוכיות זמן

המחלקת הסיבוכיות זמן מסומנת $\text{TIME}\left(t(n)\right)$ ומוגדרת להיות אוסף של כל השפות אשר ניתנות להכרעה על ידי מכונת טיורינג בזמן $O\left(t(n)\right)$.

משפט 20:

הגדרת זמן הריצה שנתנו היא תלויה במודל של מכונת הטיורינג שאיתו אנחנו עובדים.

משפט 21:

תהי $t:\mathbb{N}\to\mathbb{R}^+$ פונקציה $t(n)$.
אם מתקיים

$$t(n)\geq n$$

אז לכל מכונת טיורינג $O\left(t(n)\right)$ רב-סרטי קיימת מ"ט $O\left(t^2(n)\right)$ עם סרט אחד.

הגדרה 26: זמן הריצה של מ"ט לא דטרמיניסטית

יהי N מכונת טיורינג לא דטרמיניסטית.
הזמן הריצה של N מוגדרת להיות הפונקציה $f:\mathbb{N}\to\mathbb{N}$ כאשר $f(n)$ הוא המספר הצעדים המקסימלי אשר N מתוך כל הענפים של החישוב שלה על קלט של אורך n.

משפט 22:

תהי $t(n)$ פונקציה המקיימת $t(n)\geq n$. כל מ"ט $O\left(t(n)\right)$ לא דטרמיניסטית N סרט אחד, שקולה למכונת טיורינג $2^{O(t(n))}$ דטרמיניסטית סרט אחד.

הגדרה 27: מכונת טיורינג פולינומית

מכונת טיורינג M תיקרא **פולינומית** או **יעילה** אם קיים $c\in\mathbb{N}$ כך ש- M פועלת בסיבוכיות זמן ריצה $O\left(n^c\right)$.

הגדרה 28: המחלקה P

המחלקה P היא אוסף השפות שקיימת מכונת טיורינג פולינומיאלית M המכריעה אותן. כלומר:

$$P=\bigcup_k \text{TIME}\left(n^k\right)\ .$$

הגדרה 29: המחלקה $POLY$

המחלקה $POLY$ היא אוסף הפונקציות שעבורן קיימות מכונת טיורינג פולינומיאלית M המחשבת אותן.

הגדרה 30: מסלול המילטוני

מסלול המילטוני (Hamiltonian cycle) בגרף מכוון $G=(V,E)$ הוא מסלול אשר עובר כל קדקוד בדיוק פעם אחת.

גרף המכיל מסלול המילטוני מכונה **גרף המילטוני** (Hamiltonian). אחרת, הגרף מכונה **לא המילטוני** (non-Hamiltonian).

הגדרה 31: הבעיית מסלול המילטוני

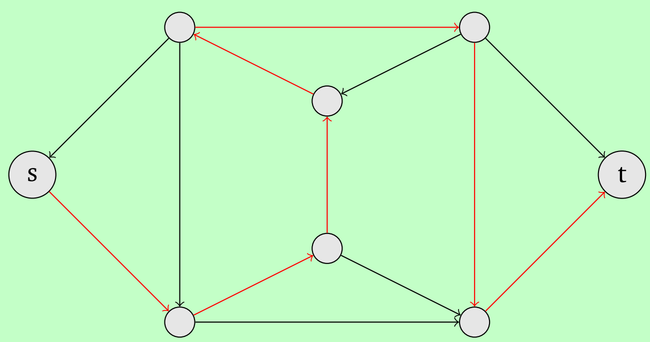
הבעיית המסלול ההמילטוני (the hamiltonian cycle problem) היא הבעיה:

" האם לגרף G יש מסלול המילטוני ?"

ניתן להגדיר כשפה פורמלית:

$$HAMPATH = \{\langle G,s,t\rangle \mid \text{הוא גרף מכוון המכיל מסלול המילטוני מ- } s \text{ ל- } t. \; G\}$$

התרשים למטה מראה דוגמה של מסלול המילטוני בגרף מכוון.



הגדרה 32: מספר פריק

מספר שלם x נקרא **פריק** (composite) אם קיימים שלמים $p>1, q>1$ כך ש-

$$x = pq \ .$$

במילים אחרות, x פריק אם ורק אם x לא ראשוני.

הגדרה 33: הבעיה $COMPOSITES$

הבעיה $COMPOSITES$ היא הבעיה:

" האם השלם x פריק? "

ניתן להגדיר כשפה פורמלית:

$$COMPOSITES = \{x \mid \text{קיימים שלמים } p,q>1 \text{ כך ש- } x=pq \}$$

הגדרה 34: אלגוריתם אימות

אלגוריתם אימות של שפה A הוא אלגוריתם V כך ש-:

$$A = \{w \mid V \text{ מקבל } \langle w,c\rangle \text{ על פי } c\}$$

במילים, **אלגוריתם אימות** הוא אלגוריתם V אשר מאמת כי הקלט w שייך לשפה A על פי התנאי c, שנקרא **אישור** (certification).

אנחנו מגדירים את זמן הריצה של V על פי האורך של w. לכן **אלגוריתם אימות זמן-פולינומיאלי** רץ בזמן פולינומיאלי $O\left(n^k\right)$ כאשר n האורך של w.

6 המחלקה NP

הגדרה 35: מחלקת הסיבוכיות NP

- המחלקה NP היא מחלקת השפות שניתן לאמתן באמצעות אלגוריתם זמן-פולינומיאלי.

הגדרה חלופית למחלקה NP הינה:

- המחלקה NP היא מחלקת השפות שניתן להכרעה באמצעות מ"ט אי-דטרמיניסטית זמן-פולינומיאלית.

למטה במשפט 23.

דוגמה 1

הוכיחו כי

$$HAMPATH \in NP \ .$$

פתרון:

כזכור, הזמן הריצה של מ"ט אי-דטרמיניסטית מוגדר לפי הזמן הריצה של הענף הכי ארוך (הגדרה 26 שלעיל).
נבנה מ"ט אי-דטרמיניסטית N_1 אשר מכריעה את $HAMPATH$ -בזמן-פולינומיאלי.

יהיו m מספר הקדקודים של G ו- n מספר הקשתות של G:

$$n = |V| \ , \qquad m = |E| \ .$$

$N_1 =$ על הקלט $\langle G, s, t\rangle$, כאשר G גרף מכוון ו- s, t קדקודים של G:

1) רושמים רשימה של n מספרים, $p_1, p_2, \ldots, p_n$.
כל מספר נבחר בצורה אי-דטרמיניסטית מ- 1 עד n.

2) בודקים אם יש חזרות ברשימה זו.
אם יש חזרות $\leftarrow$ rej.

3) בודקים אם $s = p_1$ ו- $t = p_n$.
אם לא $\leftarrow$ rej.

4) לכל $1 \leq i \leq n-1$ בודקים אם הקשת (p_i, p_{i+1}) שייכת לקבוצת הקשתות E של G.

- אם אף קשת לא שייכת ל- E ← rej.
- אם כל הקשתות שייכות ל- E ← acc.

כעת נבדוק את הסיבוכיות של האלגוריתם הזה.

- שלב 1) דורש n צעדים ולכן מתבצע בזמן פולינומיאלי.
- שלב 2) דורש n צעדים לכל היותר, ולכן מתבצע בזמן פולינומיאלי.
- שלב 3) דורש n צעדים לכל היותר, ולכן מתבצע בזמן פולינומיאלי.
- עבור שלב 4) לכל קשת (p_i, p_{i+1}), המ"ט N_1 בודקת אם יש קשת תואמת בקבוצת הקשתות E של G.
  לכן ידרשו m צעדים לכל היותר לכל i.
  לכן שלב 4) דורש $m(n-1)$ צעדים לכל היותר בסה"כ.

לכן הסיבוכיות זמן-הריצה של N_1 היא

$$O(n) + O(n) + O\left(m(n-1)\right) = O\left(m(n-1)\right)$$

לפיכך האלגוריתם הזה מתבצע אי-דטרמיניסטי בזמן פולינומיאלי.

משפט 23: $A \in NP$ אם"ם A ניתנת לאימות ע"י N_{TM}

שפה A כלשהי שייכת למחלקה NP אם ורק אם A ניתנת להכרעה על ידי מכונות טיורינג אי-דטרמיניסטית זמן-פולינומיאלית.

רעיון ההוכחה:
הרעיון הוא להראות כיצד להמיר אלגוריתם אימות זמן-פולינומיאלי למכונת טיורינג אי-דטרמיניסטית זמן-פולינומיאלית ולהפך.
במילים פשוטות אלגוריתם אימות זמן-פולינומיאלי V שקול חישובי למ"ט אי-דטרמיניסטית זמן-פולינומיאלי N_{TM}:

- N_{TM} מדמה V על ידי ניחוש של האישור c.
- V מדמה N_{TM} באמצעות המסלול של N_{TM} אשר מקבל את השפה בתור האישור.

הוכחה:
$\Leftarrow$

ראשית נוכיח שאם $A \in NP$ אז A ניתנת לאימות ע"י N_{TM}.

$A \in NP$ לכן קיים אלגוריתם אימות זמן פולינומיאלי V של A.

נבנה מ"ט N שרץ בזמן $O\left(n^k\right)$ עבור k כלשהו.

$N =$ " על הקלט w של אורך n:

1) בצורה אי-דטרמיניסטית בוחרים מחרוזת c באורך n^k לכל היותר.

נשים לב שחייב להיות חסם עליון n^k על האורך של c עבור k כלשהו, בגלל ההנחה שלנו ש- V עצמו הוא אלגוריתם זמן-פולינומיאלי.

2) מריצים V על $\langle w, c\rangle$:

3)
- אם V מקבל אז $N \leftarrow$ acc.
- אחרת $N \leftarrow$ rej."

$\Rightarrow$

נוכיח שאם A ניתנת לאימות ע"י מ"ט אי-דטרמיניסטית זמן-פולינומיאלית אז $A \in NP$.

נניח ש- A ניתנת לאימות ע"י מ"ט אי-דטרמיניסטית זמן-פולינומיאלית N.
נבנה אלגוריתם אימות זמן פולינומיאלי כמפורט להלן:

בהינתן קלט w ומכונת טיורינג אי-דטרמיניסטית N אשר מאמתת כי $w \in A$ בזמן-פולינומיאלי. נסמן ב- n את האורך של הקלט w.

ראשית הוכחנו בהפרק על מכונות טיורינג אי-דטרמיניסטיות, שכל מכונת טיורינג אי-דטרמיניסטית שקולה חישובית למכונת טיורינג דטרמיניסטית 3-סרטים:
(1) סרט הכספת, (2) סרט העבודה ו-(3) סרט הבחירות.
על סרט הבחירות המכונת טיורינג דטרמיניסטית רושמת כל הסדרות של הבחירות בסדר לקסיקוגרפי.
בנוסף מובטח לנו כי האורך של סרט הבחירות חסום מלמעלה על ידי n^k (עבור k טבעי כלשהו) מסיבה לכך שהנחנו ש- N רצה בזמן פולינומיאלי.

תהי c אחת הסדרות של הבחירות. שוב, אורך הסרט הבחיורת חסום מלעלה על יד n^k לכן גם c חסום מלמעלה על ידי n^k.
נבנה אלגוריתם אימות V כך:

$V =$ " על הקלט $\langle w, c\rangle$ כאשר w ו- c מחרוזות:

1) מריצים N על הקלט w.

V מתייחס לכל תו של c כתיאור של בחירה האי-דטרמיניסטית לבצע בכל צעד.

2)
- אם המסלול הנוכחי של החישוב של $N \leftarrow$ acc אז V מקבל את $\langle w, c\rangle$.
- אם המסלול הנוכחי של החישוב של $N \leftarrow$ rej אז V דוחה את $\langle w, c\rangle$."

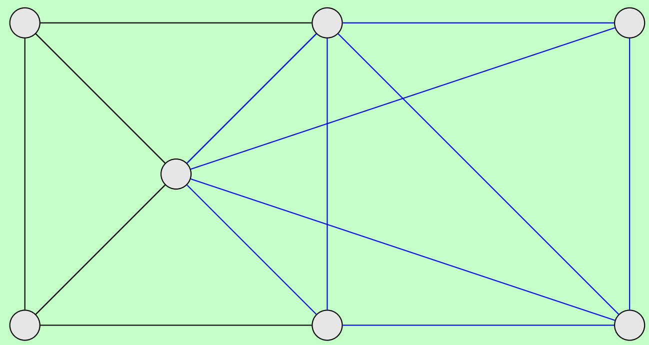
■

הגדרה 36: k-קליקה

נתון גרף בלתי-מכוון.

- קליקה בגרף בלתי-מכוון הוא תת-גרף שבו כל זוג קדקודים קשורים על ידי קשת.
- k- קליקה היא קליקה שבו יש בדיוק k קדקודים.

התרשים למטה מראה דוגמה של 5- קליקה.



דוגמה 2 בעיית הקליקה

בעיית הקליקה היא הבעיה לקבוע האם גרף מכיל k- קליקה עבור k מסוים:

$$CLIQUE = \left\{ \langle G, k \rangle \mid \text{-}k \text{ גרף בלתי-מכוון שמכיל } G \right\}$$

הוכיחו כי $CLIQUE \in NP$.

הוכחה: עבור הגרף $G = (V, E)$ יהי $n = |V|$ מספר הקדקודים ו- $m = |E|$ מספר הקשתות.

האלגוריתם הבא הוא מאמת V של $CLIQUE$:

$V =$ " על הקלט $\langle \langle G, k \rangle, c \rangle$:

(1 בודקים האם c קבוצה של k קדקודים שבגרף G.
- אם לא ← rej.
- אם כן ממשיכים לשלב 2).

(2 בודקים אם G מכיל את כל הקשתות אשר מקשרות בין כל הקדקודים ב- c.
- אם לא ← rej.
- אם כן ← acc."

- שלב 1) דורש k צעדים לכל היותר.
- שלב 2) בכל k-קליקה יש $\frac{1}{2}k(k-1) = \binom{k}{2}$ קשתות בסה"כ. לכן בשלב 2) האלגוריתם צריך לבדוק אם כל אחת של הקשתות מוכלת ב-G. ז"א לכל קשת של c האלגוריתם סורק את קבוצת הקשתות E אחת אחת ובודק אם יש קשת תואמת. לכן שלב 2) דורש $m \times \frac{1}{2}k(k-1)$ לכל היותר.

לפיכך הסיבוכיות זמן הריצה היא

$$O(k) + O(mk(k-1)) = O\left(m^3\right) \ .$$

כלומר האלגוריתם המאמתת רץ בזמן פולינומיאלי.

לכן $CLIQUE \in NP$. ■

הגדרה 37: בעיית סכום התת קבוצה SUBSET-SUM

נתונה קבוצת שלמים

$$S = \{x_1, \ldots, x_k\}$$

ושלם t. נתונה קבוצה נוצרת סופית $S \subset \mathbb{N}$ של שלמים ונתון ערך מטרה $t \in \mathbb{N}$ שלם.
בבעיית סכום התת-קבוצה SUBSET-SUM, אנחנו שואלים אם קיימת תת-קבוצה $S \subseteq Y$ כך שהאיברים שלה מסתכמים לערך t.
נגדיר את הבעיה כשפה:

$$SUBSETSUM = \left\{ \langle S,t\rangle \;\middle|\; \text{קיימת תת-קבוצה } Y \subseteq S \text{ כך שמתקיים } \sum_{y\in Y} y = t \right\}$$

לדוגמה, אם

$$S = \{1, 16, 64, 256, 1040, 1041, 1093, 1284\}$$

ו- $t = 3754$ אזי התת-קבוצה

$$Y = \{1, 16, 64, 256, 1040, 1093, 1284\}$$

היא פתרון.

דוגמה 3

הוכיחו:

$$SUBSETSUM \in NP \ .$$

הוכחה: אנחנו נבנה מ"ט זמן-פולינומיאלי M אשר מאמת פתרון כלשהו לבעיית סכום התת-קבוצה.

תהי M מ"ט דטרמיניסטית 3 סרטים:

- על סרט S רשומים האיברים של הקבוצה S בבסיס אונרי עם תו "#" להפריד בין איברים.
- על סרט c רשומים האיברים של הקבוצה c בבסיס אונרי עם תו "#" להפריד בין איברים.
- על סרט t רשום המספר t בבסיס אונרי.

לדוגמה, אם

$$S = \{1, 2, 3, 4\} \ , \qquad c = \{2, 3, 4\} \ , \quad t = 9 \ .$$

אז התכנים של הסרטים יהיו

S	␣	1	#	1	1	#	1	1	1	#	1	1	1	1	␣
c	␣	1	1	#	1	1	1	#	1	1	1	1	␣	␣	␣
t	␣	1	1	1	1	1	1	1	1	1	␣	␣	␣	␣	␣

האלגוריתם של M מתואר להלן.

$M =$ " על הקלט $\langle\langle S,t\rangle, c\rangle$:
בשלב הראשון אנחנו בודקים אם S מכילה את כל השלמים שב- c.

שלב 1) הראש S והראש c זזים ימינה צעד אחד במקביל.

- אם ראש c קורא ␣ וראש S קורא 1 ← rej.
- אם ראש c קורא 1 וראש S קורא ␣ ← rej.
- אם ראש c קורא # וראש S קורא 1,
  או אם ראש c קורא 1 וראש- S קורא #:
  - ראש c חוזר לתחילת המחרוזת
  - ראש S זז למשבצת הבאה אחרי ה- #.
- אם ראש c קורא # וראש S קורא # אז הראש S והראש c זזים שניהם משבצת אחת ימינה וממשיכים לשלב 2).

שלב 2)
- אם ראש-c קורא ␣ וראש- S קורא ␣, מחזירים ראש S וראש c לתחילת המחרוזת ועוברים לשלב 3).
- אחרת חוזרים על שלב 1)

בשלבים 3) ו- 4) אנחנו בודקים אם הסכום של האיברים של c שווה ל- t.

שלב 3) בשלב זה אנחנו מחברים את המספרים על סרט c:

עבור כל תו # בסרט c, כותבים עליו 1 ומוירדים תו 1 תואם מקצה הימין של הסרט, ומחזירים את הראש לתחילת הסרט c.

שלב 4) בשלב זה אנחנו בודקים שהמספרים על הסרים c ו- t שווים.

הראשים של c ושל t זזים ימינה צעד צעד במקביל.

- אם ראש c קורא ␣ וראש t קורא 1 ← rej.
- אם ראש c קורא 1 וראש t קורא ␣ ← rej.
- אם ראש c קורא ␣ וראש t קורא ␣ ← acc."

כעת נבדוק את הסיבוכיות של האלגוריתם. נסמן ב- n האורך המקסימלי מבין הסרטים S, c, t.

- שלבים 1) ו- 2) דורשים n^2 צעדים לכל היותר.
- שלב 3) דורש $2n$ שלבים לכל היותר.
- שלב 4) דורש n שלבים לכל היותר.

לכן

$$M = O\left(n^2\right) + O\left(2n\right) + O\left(n\right) = O\left(n^2\right)$$

לכן $SUBSETSUM \in NP$. ■

ההוכחה חלופית: נבנה מ"ט אי-דטרמיניסטית N שמכריעה את השפה SUBSET-SUM כמפורט להלן:

" $N =$ על הקלט $\langle S,t\rangle$:

1) נבחר בצורה אי-דטרמיניסטית תת-קבוצה c של השלמים S.

2) בודקים אם הסכום של האיברים של c שווה ל- t:

- אם $\sum\limits_{y\in c} y = t$ אז N ← acc.
- אם $\sum\limits_{y\in c} y = t$ אז N ← rej. "

7 NP-שלמות

עד כה אנחנו ראינו את הגדרות של המחלקות P ו- NP:

$P=$ מחלקת השפות שכריעות בזמן פולינומיאלי ,

$NP=$ מחלקת השפות שניתנות לאימות בזמן פולינומיאלי .

- ראינו שתי דוגמאות של שפות, $HAMPATH$ ו- $CLIQUE$ ששייכות ל- NP אך לא ידוע האם הן שייכות גם ל- P.
- שאלה מרכזית במדעי המחשב היא שאם $P=NP$, כלומר:
  האם כל שפה ששייכת ל- NP גם שייכת ל- P,
  וכל שפה ששייכת ל- P גם שייכת ל- NP?
  ננסח את השאלה כביטוי פורמלי. האם מתקיים

$$L\in NP \Leftrightarrow L\in P \ .$$

8 הבעיה של ספיקות

8.1 תזכורת: משתנים בוליאניים

סימן	פעולה
$\wedge$	AND
$\vee$	OR
$\neg$	NOT
$\oplus$	XOR

$0\wedge 0=0$	$0\vee 0=0$	$\neg 0=1$	$\bar{0}=1$
$0\wedge 1=0$	$0\vee 1=1$	$\neg 1=0$	$\bar{1}=0$
$1\wedge 0=0$	$1\vee 0=1$		
$1\wedge 1=1$	$1\vee 1=1$		

הגדרה 38: גרירה

יהיו p,q משתנים בוליאניים.

$p \to q = 0$ אם $p = 1$ ו- $q = 0$.

אחרת $p \to q = 1$.

הגדרה 39: אם ורק אם

יהיו
p, q משתנים בוליאניים.

$p \leftrightarrow q = 1$ אם $p = q = 0$ או אם $p = q = 1$, כלומר אם ל- p ו- q אותם ערכים.

אחרת
$$p \leftrightarrow q = 0 \ .$$

$0 \oplus 0 = 0$	$0 \leftrightarrow 0 = 1$	$0 \to 0 = 1$
$0 \oplus 1 = 0$	$0 \leftrightarrow 1 = 0$	$0 \to 1 = 1$
$1 \oplus 0 = 1$	$1 \leftrightarrow 0 = 0$	$1 \to 0 = 0$
$1 \oplus 1 = 0$	$1 \leftrightarrow 1 = 1$	$1 \to 1 = 1$

8.2 הגדרה של נוסחה ספיקה

נוסחה בווליניאית היא ביטוי במונחי משתנים בווליאניים ופעולות בוליאניות. למשל

$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z}) \ \ .$$

הגדרה 40: נוסחה בוליאנית ספיקה

אומרים כי נוסחה בוליאנית ϕ ספיקה אם קיימת השמת ערכי אמת הגורמת לכך שהערך שמייצגת הנוסחה יהיה 1.

דוגמה 4

הנוסחה
$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$$

ספיקה מסיבה לכך שקיימת השמה

$$x = 0, \quad y = 1, \quad z = 0$$

עבורה
$$\phi = 1 \ .$$

אומרים כי ההשמה $x = 0, y = 1, z = 0$ מספקת את ϕ.

דוגמה 5

נתונה הנוסחה

$$\phi = ((x_1 \leftarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

מצאו השמה מספקת ל- ϕ.

פתרון:

ההשמה

$$\langle x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1 \rangle$$

היא השמה מספקת. שכן:

$$\begin{aligned}\phi =& ((0 \leftrightarrow 0) \vee \neg((\neg 0 \leftrightarrow 1) \wedge 1)) \wedge \neg 0 \\ =& (1 \vee \neg(1 \wedge 1)) \wedge 1 \\ =& (1 \vee 0) \wedge 1 \\ =& 1 \ .\end{aligned}$$

ולכן נוסחה ϕ זו שייכת ל- SAT.

הגדרה 41: הבעיית הספיקות SAT

הבעיית הספיקות שואלת אם נוסחה בוליאנית נתונה היא ספיקה. במונחי שפות פורמלית:

$$SAT = \left\{ \langle \phi \rangle \;\middle|\; \phi \text{ היא נוסחה בוליאנית ספיקה} \right\}$$

האלגוריתם הנאיבי הקובע אם נוסחה בוליאנית כלשהי היא ספיקה אינו רץ בזמן פולינומיאלי. עבור נוסחה ϕ המכילה n משתנים קיימות 2^n השמות אפשריות. אם האורך של $\langle \phi \rangle$ פולינומיאלי ב- n, אזי בדיקה כל ההשמות דורשות זמן על-פולינומיאלי. כפי שמוכיח המשפט שלהלן, לא קיים ככל הנראה אלגוריתם זמן-פולינומיאלי עבור בעיה זו.

9 הגדרה של רדוקציה (תזכורת)

הגדרה 42: פונקיצה הניתנת לחישוב

פונקציה $f : \Sigma^* \to \Sigma^*$ ניתנת לחישוב אם קיימת מ"ט M, עבורה על הקלט w M עוצרת עם $f(w)$ על הסרט שלה.

הגדרה 43: פונקציה שניתנת לרדוקציה

השפה A ניתנת לרדוקציה לשפה B, נסמן $A \leq_m B$, אם קיימת פונקציה שניתנת לחישוב $f : \Sigma^* \to \Sigma^*$ כך שלכל

$$w \in A \Leftrightarrow f(w) \in B \ .$$

הפונקציה f נקראת הרדוקציה של A ל- B.

10 הגדרה של רדוקציה זמן-פולינומיאלית

הגדרה 44: פונקיצה הניתנת לחישוב זמן-פולינומיאלי

פונקציה $f:\Sigma^*\to\Sigma^*$ ניתנת לחישוב זמן-פולינומיאלי אם קיימת מ"ט זמן-פולינומיאלית M, עבורה על הקלט w, M עוצרת עם $f(w)$ על הסרט שלה.

הגדרה 45: פונקציה שניתנת לרדוקציה זמן-פולינומיאלית

השפה A ניתנת לרדוקציה זמן-פולינומיאלית לשפה B, שנסמן $A\leq_P B$, אם קיימת פונקציה שנתנת לחישוב זמן-פולינומיאלית $f:\Sigma^*\to\Sigma^*$ כך שלכל

$$w\in A\Leftrightarrow f(w)\in B\ .$$

הפונקציה f נקראת **הרדוקציה זמן-פולינומיאלית** של A ל- B.

משפט 24: אם $A\leq_P B$ ו- $B\in P$ אז $A\in P$

אם $A\leq_P B$ ו- $B\in P$ אז $A\in P$.

הוכחה:
$B\in P$ לכן קיימת מ"ט M_B זמן-פולינומיאלית שמכריעה את B.
$A\leq_P B$ לכן קיימת רדוקציה זמן פולינומיאלית f מ-A ל- B.

נבנה מ"ט M_A שמכריעה את A:

$M_A=$ " על הקלט w:

1) מחשבים את $f(w)$.

2) מריצים M_B על הקלט $f(w)$.

3) מחזירים את הפלט של M_B.

מכיוון ש- f רדוקציה של B ל-A אז $f(w)\in B$ אם ורק אם $w\in A$.
לכן M_B מקבלת את $f(w)$ לכל $w\in A$.

הוכחנו כי M_A מכריעה את A.

כעת נוכיח כי $M_A\in P$.
f רדוקציה זמן פולינומיאלי $\Leftarrow$ שלב 1) מתבצע בזמן פולינומיאלי.
M_B מ"ט זמן פולינומיאלית ו- f חישובית זמן-פולינומיאלית $\Leftarrow$ שלב 2) מתבצע בזמן פולינומיאלי (מכיוון שהרכבה של שני פולינומים היא פולינום).

■

11 ספיקות נוסחאות 3-CNF

הגדרה 46: ליטרל (literal)

ליטרל (literal) בנוסחה בולינאנית הוא מופע של משתנה בוליאני $x \in \{0,1\}$ או שלילתו, $\bar{x}$.

הגדרה 47: פסוקית (clause)

פסוקית (clause) היא נוסחה בוליאנית שמכילה ליטרלים שמחוברים על ידי פעולות $\vee$.
למשל

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4 \ .$$

הגדרה 48: צורה קוניונקטיבית נורמלית (CNF)

אומרים כי נוסחה בוליאנית היא **צורה קוניונקטיבית נורמלית** (conjunctive normal form) ובקיצור CNF אם היא מבוטאת כ- AND של פסוקיות שכל אחת מהן היא OR של ליטרלים אחד או יותר.
למשל

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6) \ .$$

הגדרה 49: צורה 3-CNF

נוסחה בוליאנית נתונה בצורה 3-CNF (3-conjunctive normal form) אם כל פסוקית מכילה בדיוק שלושה ליטרלים שונים.
למשל

$$(x_1 \vee \bar{x}_1 \vee \bar{x}_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$$

היא נוסחה 3-CNF. הראשונה בשלוש הפסוקיות שלה היא $(x_1 \vee \bar{x}_1 \vee \bar{x}_2)$, המכילה את שלושת הליטרלים $x_1, \bar{x}_1, \bar{x}_2$.
דוגמה נוספת של 3-CNF:

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6) \ .$$

הגדרה 50: הבעיית 3-SAT

בעיית ספיקותן של נוסחאות 3-CNF שואלת אם נוסחת 3-CNF בוליאנית נתונה ϕ היא ספיקה.
בשפה פורמלית:

$$3SAT = \left\{ \langle \phi \rangle \ \middle| \ \text{היא נוסחת 3-CNF ספיקה.} \ \phi \right\}$$

משפט 25: 3-SAT ניתנת לרדוקציה זמן-פולינומיאלית ל- CLIQUE

בהביית 3-SAT ניתנת לרדוקציה זמן-פולינומיאלית לבעיית CLIQUE:

$$3SAT \leq_p CLIQUE \ .$$

הוכחה:

תהי ϕ נוסחה בוליאנית עם k פסוקיות:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k) \ .$$

תהי R פונקצית הרדוקציה שיוצרת את המחרוזת $\langle G, k\rangle$ כאשר $G = (V, E)$ גרף בלתי-מכוון שמוגדר כמפורט להלן.

- עבור כל פסוקית $C_i = (a_i \vee b_i \vee c_i)$ נוסיף ל- V שלושה של קדקודים $T_i = (a_i, b_i, c_i)$.
- נחבר בקשת שני קדקודים v_i ו- v_j $(i, j = 1, \ldots, k)$ אם מתקיימים שני התנאים הבאים:

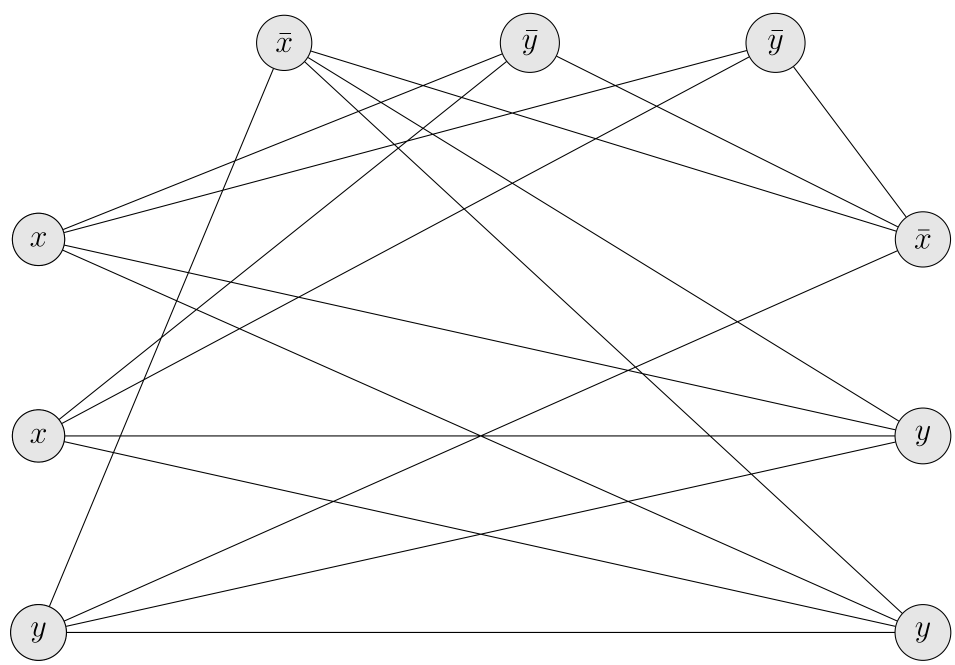
תנאי 1) v_i ו- v_j שייכים לשלושות שונות, דהיינו $i \neq j$.

תנאי 2) הליטרלים המתאימים להם **קונסיסטנטיים**, כלומר v_i אינו שלילתו של v_j.

למשל בהינתן הנוסחה

$$\phi = (x \vee x \vee y) \wedge (\bar{x} \vee \bar{y} \vee \bar{y}) \wedge (\bar{x} \vee y \vee y)$$

הגרף G אשר ה- 3CNF הזה יוצר מתואר בתרשים למטה:



כעת נוכיח כי ϕ ספיקה אם ורק אם G מכיל קליקה.

- נניח שעבור ϕ קיימת השמה מספקת.
- עבור ההשמה הזו בכל פסוקית יש לפחות ליטרל אחד אשר הוא "אמת" (שווה ל-1).
- נבחר ליטרל אשר הוא אמת בכל פסוקית.

- הקבוצת הקדקודים הנבחרים בשלב הקודם מהווים k- קליקה:

הרי אנחנו בחרנו k קדקודים, ובנוסף מובטח לנו שכל זוג קדקודים מקושרים, בגלל שכל זוג קדקודים מקיים את השני התנאים שלעיל:

תנאי 1) אף זוג קדקודים אינם מאותה שלושת מכיוון שבחרנו ליטרל אחד מכל פסוקית.

תנאי 2) אף קדקוד לא השלילתו של קדקוד השני באף זוג כי כל ליטרל שבחרנו הוא אמת.

לכן G מכיל k-קליקה.

עכשיו נוכיח שאם G מכיל k-קליקה אז ϕ ספיקה.

- נניח ש- G מכיל k-קליקה.
- ב k-קליקה זו, אין אף זוג קדקודים שבאותה שלושת בגלל שקדקודים מאותה שלושת אינם מחוברים בקשת.
- נבחר השמת ערכים לליטרלים של ϕ כך שהליטרלים שמהווים הקדקודים של ה k-קליקה הם אמת.

  ההשמה הזו תמיד אפשרית מכיוון שב- G אין זוג קדקודים בעלי ערכים משלימים שמחוברים בקשת.
- ההשמה זו מספקת את ϕ בגלל שבכל פסוקית של 3 ליטרלים יהיה לפחות ערך אמת אחד, ולכן ϕ מסופקת.

■

מסקנה 1: $3SAT \in P \quad \Rightarrow \quad CLIQUE \in P$

לפי משפט 24 ומשפט 25:

אם $CLIQUE \in P$ אז $3SAT \in P$.

12 NP שלמות

רדוקציות זמן-פולינומיאליות מספקות אמצעי פורמלי שבעזרתו אפשר להראות כי בעיה אחת קשה לפחות כמו בעיה אחרת, עד כדי גורם זמן-פולינומיאלי. כלומר, אם $A \leq_p B$ אזי B קשה יותר מ- A בגורם פולינומיאלי לכל היותר. זוהי הסיבה לכך שהשימוש בסימן "$\leq$" לציון רדוקציה מתאים.
עכשיו אנחנו נגדיר את מחלקת השפות ה- NP- שלמות שהן הבעיות הקשות ביותר ב- NP.

הגדרה 51: **NP-שלמות**

שפה B היא **NP- שלמה** או **שלמה ב- NP** (NP-complete) אם היא מקיימת את השני התנאים הבאים:

1) $B \in NP$ וגם

2) $A \leq_p B$ עבור כל $A \in NP$.

במילים פשוטות: כל A ב- NP ניתנת לרדוקציה זמן-פולינומיאלית ל- B.

הגדרה 52: **NP קשה**

אם שפה B מקיימת את תכונה 2) אולם לא בהכרח את תכונה 1) בהגדרה 51
אז אומרים כי B **NP- קשה** או **קשה ב- NP** (NP-hard).

משפט 26:

אם B NP- שלמה ו- $B\in P$ אז $P=NP$.

הוכחה:
נניח ש- B NP- שלמה. אז:

- $B\in NP$ וגם
- כל שפה $A\in NP$ ניתנת לרדוקציה לשפה B בזמן-פולינומיאלי:

$$A\leq_p B\ .$$

בנוסף נניח ש- $B\in P$. ז"א קיימת מ"ט דטרמיניסטית זמן-פולינומיאלית M שמכריעה את B.

לכל $A\in NP\ \exists$ רדוקציה חישובית זמן-פולינומיאלית R כך ש-

$$A\leq_p B\ .$$

ז"א הכרעה של B בזמן פולינומיאלי מאפשרת הכרעה של A בזמן פולינומיאלי.

- מכיוון ש- $B\in P$ אז כל שפה $A\in NP$ כריעה בזמן פולינומיאלי באמצעות הרדוקציה R זמן-פולינומיאלי והמ"ט דטרמיניסטית זמן-פולינומיאלית M.
- לכן $A\in P$ לכל $A\in NP$.

■

משפט 27: אסוציאטיביות של NP שלמות

אם השני תנאים הבאים מתקיימים:

1) B היא שפה NP- שלמה.

2) קיימת $C\in NP$ עבורה $B\leq_p C$.

אז C שפה NP- שלמה.

הוכחה:
כדי להוכיח ששפה C תהיה NP- שלמה, לפי הגדרה 51 יש להוכיח ש:

1) $C\in NP$

2) עבור כל שפה $A\in NP$ מתקיים $A\leq_p C$.

התנאי הראשון כבר נתון. נשאר רק להוכיח שתנאי השני מתקיים.

- B NP- שלמה $\xLeftarrow{\text{הגדרה 51}}$ $A\leq_p B$ לכל $A\in NP$.

(כלומר כל שפה $A\in NP$ ניתנת לרדוקציה זמן-פולינומאלית ל- B).

- בנוסף נתון כי $B \leq_P C$ לכל $C \in NP$.
- ז"א $C \leq_p B \leq_p A$ לכן $C \leq_p A$ לכל $A \in NP$.
(כלומר, רדוקציות זמן-פולינומיאלית ניתנת להרכבה).

לכן קיבלנו ש-
$$A \leq_p C$$
לכל $A \in NP$ ולכן השפה C היא NP- שלמה.

■

משפט 28: משפט קוק לוין

הבעיית SAT היא NP - שלמה.

הוכחה:
חשיפה מלאה: ההוכחה הבאה מתבססת על ההוכחה שנתונה בהספר של Sipser.

על פי הגדרה 51 יש להוכיח ששני התנאים הבאים מתקיימים:

תנאי 1: $SAT \in NP$.

תנאי 2: $A \leq_p SAT$ לכל $A \in NP$.

ראשית נוכיח כי $SAT \in NP$:

כדי להוכיח כי SAT שייכת ל- NP, נוכיח כי אישור המורכב מהשמה מספקת עבור נוסחת קלט ϕ ניתן לאימות בזמן פולינומיאלי.
נניח כי $|\phi| = n$. כלומר ב- ϕ מופיעים n ליטרלים.
ז"א השמה כלשהי דורשת n משתני בוליאניים לכל היותר.

- אלגוריתם האימות מחליף כל משתנה בנוסחה בערך המתאים לו על פי ההשמה.
השלב הזה הוא $O(n)$.
- אחר כך האלגוריתם מחשב את ערכו של הביטוי:
  * נניח כי הנוסחה ϕ מכילה k דורות של סוגריים בתוך סוגריים.
  * החישוב מתחיל עם החישובים של הביטויים בתוך הסוגריים הכי בפנים.
  * יש n סוגריים הכי-בפנים לכל היותר, וכל אחד של הסוגריים האלה מכיל n ליטרלים לכל היותר.
  לכן החישוב הזה הוא $O(n^2)$.
  * יש k דורות של סוגריים לכן החישוב כולו הוא $O(kn^2)$
- בסה"כ הסיבוכיות זמן הריצה היא
$$O(n) + O(kn^2) = O(n^2)$$
לפיכך אישור של השמה כלשהי מתבצע בזמן פולינומיאלי.

- אם ערכו של הביטוי הוא 1 הנוסחה ספיקה.

הוכחנו כי $SAT \in NP$. עכשיו נוכיח כי $A \leq_p SAT$.

תהי N מ"ט אי-דטרמיניסטית זמן-פולינומיאלית שמכריעה שפה A כלשהי בזמן $O\left(n^k\right)$ עבור k טבעי.
התרשים למטה מראה טבלה של קונפיגורציות של N.
ברשימה הבאה רשומות ההגדרות של הטבלה:

- כל שורה מראה את תוכן הסרט בשלב מסוים של מסלול אחד של N.
- בשורה הראשונה יש את הקונפיגורציה ההתחלתית.
- אנחנו מניחים כי האורך של המילה, כלומר אורך הקלט הוא n.

  הסימנים $w_1, \ldots, w_n$ מסמנים את התווים של הקלט.
- בתא הראשון בכל שורה יש #, ואחר כך רשומה הקונפיגורציה של N.

  בסוף הקונפיגורציה בכל שורה יש #.

  אחרי ה- # בקצה הימין של המילה, בכל תא יש תו רווח עד הסוף של השורה.

  התווי רווח לפני המשבצת הראשונה של הקונפיגורציה לא מופיעים בטבלה.
- האורך של כל שורה הוא בדיוק n^k תאים.
- בטבלה יש בדיוק n^k שורות לסיבה הבאה:
  - המכונת טיורינג מבצעת n^k צעדים לכל היותר.
  - בכל צעד המ"ט עוברת לקונפיגורציה חדשה.
  - בכל שורה רשומה קונפיגורציה.
  - בסה"כ יש n^k שורות עבור ה- n^k קונפיגוריות שונות האפשריות.

#	q_0	w_1	w_2	$\ldots$	w_n	$\sqcup$	$\ldots$	$\sqcup$	#
#	q_0								#
#	q_0								#
#									#

אנחנו אומרים כי טבלה שלהי היא **טבלה המקבלת** אם באחת השורות יש קונפיגורציה אשר N מקבלת אותה.

בעזרת הטבלה נתאר את הרדוקציה זמן-פולינומיאלית f משפה A כלשהי ל- SAT.

הפונקצית הרדוקציה

f מקבלת קלט w ומחזירה נוסחה $\phi = f(w)$, אשר לפי ההגדרה של פונקצית הרדוקציה, עומדת בתנאי הבא:

$$w \in A \qquad \Leftrightarrow \qquad f(w) \in SAT \ .$$

יהיו Q קבוצת המצבים ו- Γ האלפיבית של הסרט של N. נגדיר

$$C = Q \cup \Gamma \cup \{\#\} \ .$$

נסמן ב- s איבר כלשהו של C.

עבור כל תא ה- (i, j) של הטבלת הקונפיגורציות נגדיר משתנה בוליאני $x_{i,j,s}$ לכל $1 \leq i, j \leq n^k$.

המשתנה x_{ijs} מוגדר על פי התנאי

$$x_{ijs} = 1$$

אם בתא ה- ij של הטבלה יש $s \in C$. למשל, אם בתא ה- $(2, 5)$ של הטבלה מופיע התו a אז

$$x_{2,5,\mathrm{a}} = 1$$

בעוד

$$x_{2,5,\mathrm{b}} = 0 \ .$$

במובן הזה, התכנים של כל התאים של הטבלה מסומנים על ידי המשתנים של ϕ.

עכשיו נבנה נוסחה ϕ על סמך התנאי שהשמה מספקת של ϕ תהיה מתאימה לטבלה המקבלת של N.

נגדיר

$$\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{acc}} \ . \tag{1}$$

אנחנו נסביר את כל הנוסחאות ϕ_{cell}, ϕ_{start}, ϕ_{move} ו- ϕ_{acc} אחד אחד למטה.

- <u>הנוסחה ϕ_{cell}</u>

כפי שמצויין לעיל, אם המשתנה $x_{i,j,s}$ "דולק", כלומר אם $x_{i,j,s} = 1$, זאת אומרת שיש סימן s בתא ה- ij של הטבלה. אנחנו רוצים להבטיח שהשמה כלשהי בנוסחה אשר מתאימה לקונפיגורציה של הטבלה, מדליקה בדיוק משתנה אחד לכל תא של הטבלה. למטרה זו נגדיר ϕ_{cell} כך:

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i,j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\overline{x}_{i,j,s} \vee \overline{x}_{i,j,t}) \right) \right] \tag{2}$$

* האיבר הראשון בסוגריים מרובעים, $\bigvee_{s \in C} x_{i,j,s}$ מבטיח שלכל תא של הטבלה, לפחות משתנה אחד דולק.
* האיבר השני $\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\overline{x}_{i,j,s} \vee \overline{x}_{i,j,t})$ מבטיח שעבור כל תא של הטבלה, משתנה אחד לכל היותר דולק.

לפיכך כל השמה מספקת עומדת בתנאי שיהיה בדיוק סימן אחד, s, בכל תא של הטבלה.

- <u>הנוסחה ϕ_{start}</u>

נוסחה ϕ_{start} מבטיחה ששורה הראשונה של הטבלה היא הקונפיגורציה ההתחלתית של N על הקלט w:

$$\begin{aligned}\phi_{\text{start}} =& x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \\ & \wedge \ldots \wedge \\ & x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned} \tag{3}$$

- <u>הנוסחה ϕ_{acc}</u>

הנוסחה ϕ_{acc} מבטיחה שקיימת טבלה קונפיגורציה אשר המ"ט N מקבלת אותה.

בפרט ϕ_{acc} מבטיחה שהסימן q_{acc} מופיע בתא אחד של הטבלה דרך התנאי שלפחות אחד המשתנים $x_{i,j,q_{\text{acc}}}$ דולק:

$$\phi_{\text{acc}} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{\text{acc}}} \tag{4}$$

- <u>הנוסחה ϕ_{move}</u>

הנוסחה ϕ_{move} מבטיחה שכל שורה של הטבלה היא "שורה חוקית".

כלומר בכל שורה, הקונפיגורציה היא כך שאפשר להגיע אליה על ידי תזוזה חוקית של N מהקונפיגורציה הקודמת שמופיעה בשורה אחת למעלה.

תזוזה חוקית בין כל שתי קונפיגורציות נקבעת על ידי הפונקצית המעברים של המ"ט N.

בשפה פורמלית, אם c_i הקונפיגורציה של שורה i, ו- c_{i+1} הקונפיגורציה של השורה $i+1$ אחת למטה, אז ϕ_{move} מבטיחה כי לכל $1\le i\le n^k-1$ מתקיים

$$c_i \vdash_N c_{i+1} \ .$$

במונחי הטבלה, אפשר להגדיר תזוזה חוקית בין כל שתי שורות על ידי תת-טבלה מסדר 2×3 שמכילה 3 תאים מתאימים של שתי שורות שכנות.

מכאן ואילך אנחנו נקרא לתת-טבלה כזאת "חלון".

למטה יש דוגמאות של חלונות חוקיים:

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	b	b
c	b	b

החלונות האלה למטה הם דוגמאות לחלונות לא חוקיים:

a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

הנוסחה ϕ_{move} קובעת שכל חלון של הטבלה חוקי. בפרט, כל חלון מכיל 6 תאים. לכן ϕ_{move} קובעת שהתכנים של ה-6 תאים של כל חלון מהווה חלון חוקי. ז"א

$$\phi_{\text{move}} = \bigwedge_{\substack{1\leq i\leq n^k\\ 1\leq j\leq n^k}} (\text{חלון ה- } i,j \text{ חוקי}) \tag{5}$$

אנחנו מציבים בטקסט " חלון ה- i,j חוקי " את הנוסחה הבאה, כאשר $a_1,\ldots,a_6$ מסמנים את התכנים של ה-6 תאים של כל חלון:

$$\bigvee_{\substack{\{a_1,a_2,a_3,a_4,a_5,a_6\}\\ \text{חלון חוקי}}} \left(x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}\right) \tag{6}$$

עד כה הוכחנו שקיים רדוקציה מכל שפה $A \in NP$ ל- SAT.
כעת נוכיח כי הרדוקציה זו חישובית בזמן-פולינומיאלי.

הטבלה של N היא מסדר $n^k \times n^k$ ולכן היא מכילה n^{2k} תאים.

נחשב את הסיבוכיות של כל הנוסחאות ϕ_{cell}, ϕ_{start}, ϕ_{acc}, ϕ_{move}.

- <u>הנוסחה ϕ_{cell}</u>

  הנוסחה (2) של ϕ_{cell} מכילה n^{2k} נוסחאות עם 3 ליטרלים. לכן
  $\phi_{\text{cell}} = O\left(n^{2k}\right)$.

- <u>הנוסחה ϕ_{start}</u>

  הנוסחה (3) של ϕ_{start} מכילה בדיוק n^k ליטרלים. לכן
  $\phi_{\text{start}} = O\left(n^{k}\right)$.

- <u>הנוסחה ϕ_{acc}</u>

  הנוסחה (4) של ϕ_{acc} מכילה בדיוק n^k ליטרלים. לכן
  $\phi_{\text{acc}} = O\left(n^{k}\right)$.

- <u>הנוסחה ϕ_{move}</u>

  הנוסחה (6,5) של ϕ_{move} מכילה n^{2k} נוסחאות עם 6 ליטרלים. לכן
  $\phi_{\text{move}} = O\left(n^{2k}\right)$.

לכן בסה"כ

$$\phi = O\left(n^{2k}\right) + O\left(n^{k}\right) + O\left(n^{k}\right) + O\left(n^{2k}\right) = O\left(n^{2k}\right) .$$

לפיכך קיימת רדוקציה חישובית בזמן פולינומיאלי מכל שפה $A \in NP$ ל- SAT.

■

משפט 29: 3-SAT היא NP שלמה.

3-SAT היא NP שלמה.