מודלים חישוביים – דף נוסחאות

1 הבונה החרוץ ואחרות

bb(n) - פונקציה <u>לא חשיבה</u> המקבלת n אורך של תוכנית ומחזירה את המספר הגדול ביותר שנית | לחשב עם תוכנית באורך n לכל היותר. **הוכחת אי חשיבות**: נניח בשלילה שקיימת תוכנית B שמחשבת את bb, ונסמן N=|B|. נכתוב תוכנית חדשה: c():=B(10*N)+1. נשים לב: מצד אחד, |c|=|B|+|N|+const=N+logN+const≤2N+const≤10N. לכן, לפי הגדרת הבונה החרוץ bb(10N)≥c(). מצד שני, לפי הגדרת c, bb(10N)<c(). קיבלנו סתירה.

$mm(n)=\min\{ l \mid l\in\mathbb{N}, \forall p\in L_0, |p|\le n, [p()]\neq l\}$ – המס' הקטן ביותר שלא ניתן לחישוב עם תוכנית באורך≤n.

$minProg(n) = \min\{|p| \mid p\in L_0, [p()]>n\}$ – אורך התוכנית המינימלית המחשבת מספר הגדול מ-n.

2 מכונת מונים (CM) ו-STS

מכונת מונים: פעולות קיימות במכונה: ++ (קידום ב-1), -- (חיסור ב-1, אם 0 נשאר 0), =0 (האם שווה ל-0). *תכונות*: <u>זיכרון</u> מספר סופי של מונים אינסופיים. <u>תוכנית</u>: מספר סופי של רצף הוראות.
<u>קלט/פלט</u>: מתבצע ברגיסטר הראשון <u>בלבד</u>. שאר המונים מאופסים. פעולת עצירה = סוף תוכנית.
<u>שיטות לתרגום רגיסטרים</u>: (א) ייצוג של כל הרגיסטרים כמספר יחיד באמצעות הראשוניים ,,, $Z=2^{m1}3^{m2}$.
(ב) הפיכת כל 2 מספרים למספר 1 וכל 2 זוגות לזוג חדש וכך הלאה $pair(x,y)=2^x(2y+1)$. כדי לקבל חזרה, בודקים כמה פעמים ניתן לחלק ב-2 – זהו x. מהמספר הנותר מורידים 1 ומחלקים ב-2 ומתקבל y.
STS מכיל: Q – מצבים, I⊆Q – קלט, O⊆Q – פלט, ז: Q→Q – פונקצית מעברים.

3 שפות תכנות

ניתן לחלק שפות תכנות ל-2 חלקים: שפות <u>מלאות</u> עבורן בעיית העצירה לא כריעה אך ניתן לכתוב מפרש לשפה בשפה ושפות <u>עוצרות</u> עבורן בעיית העצירה קריאה אבל לא ניתן לכתוב מפרש לשפה בשפה עצמה.

4 משפט רייס

בעיה טריוויאלית – היא בעיה שהתשובה עליה היא תמיד 'כן' או תמיד 'לא'.
בעיה סמנטית – היא בעיה שבהינתן 2 תוכניות שקולות (מחזירות אותו ערך לכל קלט), מתקיים P תחזיר עבור שתיהן את אותה התשובה. כלומר, f≡g ⇒ P(f)=P(g).

תנאי משפט רייס: (1) זו שפה של תוכניות (הקלט והפלט היא תוכנית). (2) הבעיה <u>אינה</u> טריוויאלית. (2) הבעיה סמנטית. [(4) הבעיה <u>אינה</u> תלויה רק במספר הקלטים.]
טענת המשפט: בהתקיים התנאים על בעיה P, אזי P אינה כריעה.
שלבי הוכחת המשפט: מחלקים את עולם הבעיות ל-2 טיפוסים a ו-b (בעיה היא מסוג a כאשר P(⊥)=F). P לא טריוויאלית לכן, קיימת תוכנית y כך ש-P(y)=T (בבעיות מסוג b: P(y)=F). מראים רדוקציה:

$$halt(f) := P\left(\lambda x.\begin{cases} yea(y) & f()=f() \\ yea(y) & o/w \end{cases}\right)$$

f עוצרת ⇐ g≡y ⇐ halt(f)=P(g)=P(y)=T.
f לא עוצרת ⇐ g≡⊥ ⇐ halt(f)=P(g)=P(⊥)=F

5 משפט הרקורסיה

לכל תוכנית <u>שמקבלת תוכנית ומחזירה תוכנית</u> קיימת נקודת שבת. כלומר קיים c≡f(c).

הוכחה לתוכניות שרצות בסדר <u>נורמלי:</u>
Let c = k(k) where k = λw.f(w(w))
c ≡ k(k) ≡ (λw.f(w(w)))(k) ≡ f(k(k)) ≡ f(c)

שלבי הוכחה לתוכניות בסדר <u>אפליקטיבי:</u>
Let c := h(k) where
k(w) := f(h(w))
h(y) := λz.y(y)(z)

6 בעיות לא כריעות

Pconst(p) - מוחזר T אם קיים n כך שלכל x, p(x)=n או p(x)=⊥.
eqv(g,h) - ∀x.[g(x)≡h(x)]
$halt_0(f)$**=same(**λc.check(f,c), λc.T) – האם 2 תוכניות שעוצרות, אי פעם מחזירות אותו ערך עבור אותו x.
Halt(f,x)**=**not(**eqv(**λn.stop(f,x,n), λc.F)) – האם 2 תוכניות שעוצרות, תמיד מחזירות אותו ערך.

halt(f,x)	f(x)≠⊥
$halt_0(f)$	f()≠⊥
halt*(f)	∀x.[f(x)≠⊥]
loop(f)	∃x.[f(x)=⊥]
empty(f)	∀x.[f(x)=⊥]

7 מנייה רקורסיבית (RE)

הטענות הבאות שקולות: (1) שפה ניתנת למנייה רקורסיבית. כלומר, קיים מונה רקורסיבי (על אבל לא בהכרח חח"ע). (2) כריעה למחצה. (3) קיימת תוכנית שמדפיסה את כל האיברים בשפה.
הערות: (א) לא כל קבוצה בת מניה ניתנת למספור רקורסיבי. יש קבוצות כמו קבוצת התוכניות שאינן עוצרות שהיא בת מנייה אך לא ניתנת לתכנות מניה רקורסיבי. **(ב)** אם L היא שפה הכריעה למחצה וגם המשלימה שלה כריעה למחצה אז L היא שפה כריעה. **(ג)** אינומרטור לשפה L היא פונקציה חשיבה ועל f:N→L.
<u>**טענה:**</u> L∈R אינסופית אמ"מ יש ל-L אינומרטור מונוטוני. כלומר, אם i>j אז f(i)>(fj) בסדר לקסיקוגרפי.
<u>**טענה:**</u> לכל שפה אינסופית L ב-RE, קיימת תת שפה אינסופית 'L כך ש-'L כריעה.

8 הוכחת R/RE/Co-RE

כיצד להראות <u>כריעות</u>? (1) אלגוריתם שפותר את הבעיה. (2) להראות שהבעיה היא "סופית". (3) רדוקציה <u>**אל**</u> בעיה הידועה ככריעה. (4) ע"י הוכחה שהיא והמשלימה שלה כריעות למחצה.
בעיות סופיות: (1) בעיות עם מספר סופי של שאילתות הן תמיד כריעות. (2) בעיות שמספר השאילתות בהם מוחזר כן/לא (מספיק אחד מהם) הוא סופי.
כיצד להראות <u>אי-כריעות</u>? (1) באמצעות שיטת האלכסון (כמו בבונה החרוץ/בעיית העצירה). (2) רדוקציה <u>**מ**</u>בעיה הידועה כלא כריעה. (3) משפט רייס. (4) ע"י חיקוי הוכחת משפט רייס.
כיצד להראות <u>כריעות למחצה</u>? (1) אלגוריתם המכריע את הבעיה למחצה. (2) רדוקציה <u>**אל**</u> בעיה הידועה ככריעה למחצה. (3) ע"י הרצת שני מונים במקביל.
כיצד להראות <u>co-RE</u>? (1) באמצעות שיטת האלכסון (כמו בSELF). (2) רדוקצית מיפוי <u>**מ**</u>בעיה הידועה כ-co-RE (כמו *halt). (3) להראות שהמשלים שלה הוא RE ושזו אינה בעיה כריעה.

9 סבוכיות קולמגורוב

הגדרה: $K(x)$ מחשבת את האורך המינמאלי של תוכנית שמחזירה את x.
מספר עובדות תחילה: ברור ש $K(x) \le |x|$ כי אפשר לכתוב את המספר. בנוסף ברור שאין משמעות לשפה שבה אנו עובדים מכיוון שבין כל שפה לשפה קיים מפרש ואורכו קבוע ונסמנו ב i אם כך לדוגמא $K_{scheme}(x) \le K_{C++}(x) + i$. ישנם מילים שלא ניתן לדחוס ועבורם $K(x) > |x|$.
מספר דוגמאות: $K(xx) \le K(x) + c$, $K(1^n) \le \log(n) + c$, $K(1^{2^n}) \le \log(n) + c$.
אבל מחרוזות עם פתרון פשוט יחסית כזה הם נדירות. הפונקציה *K(.)* מוגדרת לכל מחרוזת ולא חסומה. אבל האם היא חשיבה? <u>**טענה:**</u> הפונקציה לא חשיבה. <u>**הוכחה**</u>: נניח בשלילה שהיא כן חשיבה. אם כך לכל $n \in N$ נגדיר את y_n להיות המחרוזת y הראשונה מבחינה לקסיקוגרפית שמקיימת $n < K(y)$. אם כך יש לנו את הסדרה $\{y_n\}_{n=1}^{\infty}$ מוגדרת היטב. כיוון ש *K* חשיבה זה אומר שלכל *n* קיים קבוע *c* שעבורו $K(y_n) \le \log(n) + c$. ואם כך קיבלנו סתירה לאופן שבה הוגדר *y* ומכאן הפונקציה לא חשיבה.

10 מכונת טיורינג TM

$M=(Q,\Sigma,,\Gamma,q_0,q_a,q_r,\delta)$ כאשר Q הוא מספר סופי של מצבים, Σ א"ב סופי שעל הסרט (ללא רווח). Γ א"ב סופי של המכונה (מכיל את הא"ב של הסרט ומכיל גם רווח). q_0,q_a,q_r – מצב התחלה, מקבל ודחייה. δ - פונקצית מעבר. <u>ייצוג מצב של TM</u>: מתבצע ע"י מחרוזת xyz. כאשר x מכיל את הקלט עד המקום בו נמצא הראש, y מכיל את המצב הנוכחי ו-z מכיל את סרט הקלט מהראש ועד הסוף. לדוגמה $1011q_7011$ אומר שאנו במצב q_7, לפני הראש מופיע על הסרט 1011 ואחריו 011 (הראש נמצא על 0).
שני סוגים של TM: **Acceptor** – מקבל קלט על הסרט (עד הרווח הראשון), מקבל את הקלט כשנכנס למצב קבלה, <u>ללא פלט</u>. **Computer** – קלט על הסרט (עד הרווח הראשון), מסיים ברגע שנכנס למצב halt. פלט על הסרט (עד הרווח הראשון).
<u>סימולציה</u>: כל TM ניתן לסמלץ בסקים B_m:=λz.tm([m],ε,x,q_0). where [M] is the transition function of M as a list and strings are lists of symbols.

כמה אפשרויות קיימות למכונת טיורינג עם 3 סרטים (קלט – קר יאה בלבד, טיוטה – קריאה וכתיבה ופלט לכתיבה בלבד)**?** $|Q|*S*|\Gamma|^S*N*|\Sigma|^n$ (משמאל לימין: כמות הקלטים האפשרית, מיקום הראש בקלט, מה שכתוב על סרט הטיוטה, מיקום ראש הטיוטה, המצב הפנימי).
ב-*QSAT* כאשר אורך הסרט מוגבל, מספר הקונפיגורציות הוא $|Q|x|\Gamma|^n xn$ (מצבים, תוכן הסרט, מיקום הראש) כאשר $|x|=n$ אורך הקלט.

11 מכונת RAM

תכונות RAM: <u>זכרון פנימי</u>- סדרה סופית ובלתי חסומה של אוגרים. <u>זכרון חיצוני</u> – רצף בלתי חסום של אוגרים. <u>תוכנית</u>: רצף סופי של הוראות ממוספרות. <u>קלט/פלט</u> – ברגיסטר הראשון. קיימת פקודת עצירה.

x := 2y
```
u := y
x := 0
if u=0 then go to c else go
to b
b: u--; x++; x ++
if u=0 then go to c else go
to b
c... :
```

x := |_y / 2_|
```
t := y
x := 0
a: t--
if t=0 then go to c else go to
b
b: t--; x++; goto a
c: ...
```

x := y mod 2
```
t := y; x := 0
if t=0 then goto c else goto a
a: t --
if t=0 then goto b else goto d
d: t --
if t=0 then goto c else goto a
b: x ++
c... :
```

12 שקילות כוח החישוב

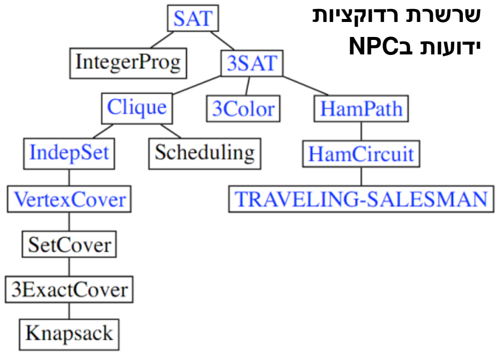
סקים ≤ TM – כותבים משערך. **TM ≤ CM** – קידוד מונה כרצף של אחדות. **CM ≤ RAM** – קידוד RAM כרצף של מספרים. **≤ RAM סקים** – שימוש במחסנית כדי לדמות מיני סקים.

- $TM_2 \leq TM$ [1 tape; 2 channels]
- $CM_2 \leq TM_2$ [111...1BBB...]
- $CM_n \leq CM_2$ [$2^i3^j5^k7^{l...}$]
- $RAM \leq CM_n$ [$2^x(2y+1)$]
- Scheme ≤ RAM [Abelson & Sussman]
- TM ≤ Scheme [Interpreter]

13 סבוכיות זמן

NP – הרמז צריך להיות בגודל פולינומיאלי והוידוי גם צריך להיות בזמן פולינומי אלי (כל זה בTM).
<u>**טענה**</u>: אם יש TM עם הרבה סרטים הפותרת בעיה ב-t(n), אם נגביל את מספר הסרטים לסרט אחד, זמן הריצה הוא לכל היותר $t^2(n)$ (במכונה עם הסרט היחיד).
<u>**טענה**</u>: ההפרש בין מודלים "סבירים" הוא רק פולינומיאלי.
NP-Hard – בעיה שמכל בעיה אחרת ב-NP ניתן להגיע אליה ע"י טרנספורמציה פולינומיאלית.
NP-Complete – בעיה שהיא NP-קשה וגם שייכת ל-NP.
<u>**טענה**</u>: G=(V,E) לא מכוון. נאמר ש-S⊆V הוא **IS** אמ"מ V/S הוא **VC**.

שרשרת רדוקציות ידועות בNPC



<u>**חשוב**</u>: אם תוכנית רצה בזמן שאינו תלוי בקלט שלה, אז היא פולינומיאלית!
CNF **-** דרך לכתוב נוסחאות. יש קבוצות ביטויים שמחוברות בקשר וגם ובתוך כל קבוצת ביטויים בין הליטרלים יש קשר או, הליטרל עצמו יכול להיות גם תחת שלילה.
CSP - בעיית הספיקות של נוסחאות עם או,וגם,שלילה ושווה. בהינתן נוסחה האם קיימת לה הצבה שעבורה התוצאה היא אמת. הבעיה ב *NPC*.
SAT - בעיית הספיקות של נוסחאות עם או,וגם,ושלילה. בהינתן נוסחה האם קיימת לה הצבה שעבורה התוצאה היא אמת. הבעיה ב *NPC*.
3SAT - בעיית הספיקות של נוסחאות מהצורה של *3CNF* מה שאומר שהם מהצורה של הרבה קבוצות ביטויים שמחוברות בקשר וגם. כאשר כל קבוצה היא 3 ליטרלים מחוברים בקשר או. הליטרל עצמו יכול להיות גם תחת שלילה. הבעיה ב *NPC*.
2SAT - בעיית הספיקות של נוסחאות מהצורה של *2CNF* מה שאומר שהם מהצורה של הרבה קבוצות ביטויים שמחוברות בקשר וגם. כאשר כל קבוצה היא *2* ליטרלים מחוברים בקשר או. הליטרל עצמו יכול להיות גם תחת שלילה. הבעיה ב *P*.
QBF - בעיית הספיקות של נוסחאות עם או,וגם,שלילה, קיים ולכל. בהינתן נוסחה האם קיימת לה הצבה שעבורה התוצאה היא אמת. הבעיה ב *PSPACE*.

14 רדוקציות

רדוקציה רגילה: $A \leq B$. בהינתן ש B חשיבה ניתן לתכנת בעזרה פונקציה חשיבה ל A.
(1) אם $B \in R$ אז $A \in R$. (2) אם $A \notin R$ אז $B \notin R$.
רדוקציית מיפוי: $A \leq_m B$. קיימת פונקציה חשיבה f כך ש $x \in A$ אמ"מ $f(x) \in B$.
(1) אם $A \leq_m B$ אז $\bar{A} \leq_m \bar{B}$. (2) אם $B \in RE$ אז $A \in RE$. אם $A \notin RE$ אז $B \notin RE$.
(3) אם $A \notin CoRE$ אז $B \notin CoRE$. אם $A \notin R$ אז $B \notin R$.
רדוקציה פולינומיאלית: $A \leq_p B$. קיימת פונקציה חשיבה ופולינומאלית f כך ש $x \in A$ אמ"מ $f(x) \in B$.
(1) אם $B \in P$ אז $A \in P$. אם $A \notin P$ אז $B \notin P$. (2) אם $B \in NP$ אז $A \in NP$. אם $A \notin NP$ אז $B \notin NP$.

<u>**משפט**</u>: אם $A \in NP-Hard$ ו $A \leq_p B$ אז $B \in NP-Hard$.
<u>**משפט**</u>: אם $A \in NPC$ $B \in NP$ ו- $A \leq_p B$ אז $B \in NPC$.

15 אוטומטים ורגולריות

אוטומט סופי דטרמניסטי (DFA)
Q – קב' סופית של מצבים
Σ – א"ב קלט סופי
δ – פונקצית המעברים (QxΣ-->Q)
q0 – מצב התחלתי
F – קב' מצבים מקבלים (מוכל ב-Q)

אוטומט סופי לא-דטרמניסטי (NFA)
כמו אוטומט דטרמניסטי עם השינויים הבאים:
Σ_ε – א"ב קלט סופי (כולל ε)
δ – פונקצית המעברים ($Qx\Sigma_\varepsilon$-->P(Q))
בפונ' המעבר כל מעבר הוא לקבוצת מעברים.
מבחינת כוח חישוב – **שקול ל-DFA** (זהים בשפות).

שפה היא רגולרית אם יש DFA שמקבל אותה. <u>**משפט**</u>: שפה רגולרית אמ"מ יש ביטוי רגולרי המתאר אותה.

הערות: (1) שפה סופית היא שפה רגולרית. (2) <u>באוטומטים</u> השפה L(A)≠?ϕ היא בעיה כריעה, כי ניתן לבדוק אם יש מצב מקבל ולוודא שקיים מסלול ממצב התחלה למצב המקבל.

הפיכת אוטומט אי דטרמיניסטי לדטרמיניסטי (עולה 2^n – עלות מערכית):
שלב 1: <u>נפטרים מכל מעברי אפסילון</u>.נניח שאני במצב A ואפסילון מעביר אותי למצב B וממצב B s מעביר אותי למצב C.אז באוטומט ביניים שלי יהיה לי מעבר ממצב A למצב C ע"י s. כך נטפל בכל מעברי האפסילון באוטומט ונקבל אוטומט ביניים, כעת נעבור לשלב הבא.
שלב 2: <u>מעבר לאוטומט דטרמיניסטי</u>. כעת כל מצב יהיה לנו קבוצה של מצבים שאליה יהיה ניתן להגיע. נניח שבאוטומט הביניים שלנו יש שני מעברי s ממצב A, אחד למצב B ואחד למצב C. אז כעת יהיה לנו באוטומט הדרמיניסטי שלנו מעבר מ A ל B,C ע"י s. מכל קבוצת מצבים שיש לנו נבדוק מה המעברים האפשריים באוטומט הביניים וככה נתקדם. נעבור על כל הריצות האפשריות במקביל וכך נבנה לנו אוטומט דטרמיניסטי.
מצבים מקבלים : מצבים שבקבוצת מצבים שלהם מכילים מצב שבמקור הוא מקבל.

למת הניפוח לרגולריים: לכל שפה רגולרית קיים מספר n (מס' המצבים באוטומט) כך שלכל מלה w המקיימת |w|>=n יש פירוק w=xyz המקיים |xy|<=n ו-1=<|y| מתקיים לכל i: $w' = xy^iz$ ששייך לשפה.

שלילת למת הניפוח לשפות רגולריות: (1) נניח בשלילה ש L רגולרית ו P קבוע הניפוח שלה. (2) נמצא WeL ו P =< |w| ונראה שלכל חלוקה w=xyz, XY<=P קיים i כך ש $XY^iZ\notin L$ בסתירה ללמת הניפוח.

סגירות רגולריות: <u>איחוד</u> – ניתן להוסיף מצב חדש עם מעברי אפסילון. <u>שרשור</u> – חיבור מצבי הסיום של הראשון עם מצב התחלה של השני במעבר אפסילון. <u>משלים</u> – באוטומט **דטרמיניסטי** נהפוך את כל המצבים המקבלים ללא מקבלים ולהפך. <u>חיתוך</u> – ע"פ דה-מורגן (בהשתמש באיחוד ומשלים). בניית חיתוך עולה 2^{l+m}. יש בניה זולה יותר (וישירה) בגודל l*m. <u>קלין סטאר *</u> - מוסיפים ממצבי הסיום מעבר אפסילון למצב התחלה חדש העובר למצב ההתחלה הישן באמצעות אפסילון. מצב ההתחלה החדש הוא מצב מקבל. <u>הפרש</u> – שפות רגולריות סגורות על הפרש.

<u>שקילות שפות רגולריות</u>: B,A אוטומטים. הבעיה L(A)=L(B) היא בעיה כריעה כי L(A)=L(B) ⇔ L(B)⊆L(A)∧L(A)⊆L(B) ⇔ $L(A)\cap\overline{L(B)}=\emptyset$. ראינו סגירות לפעולות הללו וזו בעיה כריעה לכן ניתן לבדוק

<u>אוטומט מינימאלי</u>: בעיית מציאת אוטומט מינימאלי לשפה רגולרית היא כריעה (ניתן לעבור על כל האוטומטים לפי גודל ולבדוק שקילות עד שלכל היותר מגיעים לאוטומט שעבורו מחפשים מינימאלי).
(*) ניתן למצוא גם אוטומט מינימאלי בצורה יעילה ע"י מחיקת מצבים לא נחוצים (כאלו שלא ניתן להגיע ממצב התחלה אליהם או מהם למצב סיום) ולאחר מכן, זריקת מסלולים כפולים.
(*) 2 אוטומטים מינימאליים לשפה הם זהים עד כדי שמות מצבים.

בשפות רגולריות הבעיות הבאות כריעות:
Membership $(x\in L(A))$. *Emptiness* $(L(A)=\phi)$. *Fullness* $(L(A)=\Sigma^*)$.
Eqv $(L(A)=L(B))$. *Subset* $(L(A)\subseteq L(B))$.

16 דקדוק חסר הקשר

דקדוק חסר הקשר הוא רביעייה : (V, Σ, R, S). V – קבוצת המשתנים (אותיות גדולות). Σ – קבוצת הטרמינלים (א"ב של המחרוזת הסופית). R – קבוצה <u>סופית</u> של חוקי גזירה. S – המשתנה (מצב) התחלתי.
שפה חסרת הקשר: L(G)={w| S⇒w} (ניתן לגזור מ-S את w).

הצורה הנורמלית של חומסקי: (1) רק S (התחלת) יכול לעבור ל-ε. (2) S לא יופיע בצד ימין של גזירה. (3) מעבר לטרמינלים יתבצע לבד (לא בצמוד למשתנה). (4) כל גזירה תבוצע ל-2 משתנים (בדיוק) או טרמינל בודד אחד.
אופן הביצוע: (1) הוספת מצב חדש S0 שמפנה ל-S המקורי. (2) ביטול כל חוקי A->ε (כל חוק כזה חוזרים אחורה למי שמפנה ל-A ומוסיפים גזירה ללא A. (3) ביטול כל גזירות יחידות A->B (מבטלים את ההפניה ל-B, ובמקומה רושמים ב-A את כל האפשרויות של B). (4) ביטול גזירות של 3 ומעלה. כל גזירה A->BCD הופכים לכללים חדשים : A->BA1 וכן : A1->CD. כמו כן, אם יש A->aB, אזי נהפוך ל- A->A1B כאשר A1->a. כך יש בדיוק חוקים כפולים (גזירה ל-2 משתנים).

דקדוק לינארי A->x, A->xBy **לינארי ימני** A->a, A->aB. משפחת השפות הנוצרות ע"י דקדוקים לינאריים ימניים\שמאליים היא משפחת השפות הרגולריות. נשתמש בדקדוק לינארי כדי **לייצג אוטומט כדקדוק ח"ה:**
(1) q_0 נהפוך למשתנה גזירה ראשוני R_0. (2) $\delta(q_i,a)=q_j \Leftarrow R_i\to aR_j$. (3) לכל מצב מסיים q_f נגדיר $R_f\to\varepsilon$.

סגירות ח"ה: ח"ה סגורות לאיחוד ($S_9\to S_A\,|S_B$), קלין סטאר ($S\to\varepsilon|SS$), שרשור ($S_9\to S_AS_B$), חיתוך עם שפה רגולרית, היפוך. **אינן סגורות** תחת חיתוך ומשלים.
הערה: כיוון שאין סגירות תחת המשלים, הבעיה "**האם קיימת מילה שאינה שייכת לשפה**" <u>אינה כריעה</u>.

למת הניפוח לח"ה: תהי *L* שפה ח"ה אזי קיים קבוע *p>0* כך שלכל מילה $w\in L$, $|w|\ge p$ יש פירוק $w=uvxyz$, כך ש: **(1)** $|vy|\ge 1$ **(2)** $|vxy|\le p$ **(3)** לכל $i\in N$ $uv^ixy^iz\in L$.
הנחת הבסיס איתה מוכחים את הלמה: אם מספר המשתנים הוא d ובכל צעד אנו יכולים להתפצל ל-c אפשרויות לכל היותר, הרי המילה הארוכה ביותר ללא חזרות היאא לכל היותר c^d.

שלילת למת הניפוח לח"ה: נניח בשלילה ש L ח"ה ו p קבוע הניפוח שלה. נמצא מילה w כך ש $w\in L$, $|w|\ge p$ ונראה שלכל חלוקה $w=uvxyz$, כך ש - $|vy|\ge 1, |vxy|\le p$ קיים i כך ש $uv^ixy^iz\notin L$.

בשפות ח"ה הבעיות הבאות כריעות: ***Membership*** $(x\in L(G))$. ***Emptiness*** $(L(G)=\phi)$.
<u>האם המילה שייכת לשפה של דקדוק</u> – נהפוך לצורת חומסקי (ואז אין מחיקות) ונבדוק את כל אפשרויות הגזירה עד האורך של המילה הספציפית |*x*|.
<u>האם יש דקדוק שלא נותן כלום</u> – נמחק משתנים מיותרים (לפי חומסקי) ואם קיבלנו שמשתנה ההתחלה מיותר אז השפה ריקה (אפשר גם למצוא חסם למילה הקצרה ביותר לפי למת הניפוח).
בשפות ח"ה הבעיה הבאה <u>לא</u> כריעה: ***Fullness*** $(L(G)=\Sigma^*)$.

פישוט דקדוקים: (1) סילוק משתנים שלא ניתן להגיע מהם למילה טרמינלית. (2) סילוק סימנים שלא ניתן להגיע אליהם ממשתנה ההתחלה.

17 דוגמאות

האם בעיות ב-*P* או ב-*NP*:

1. **האם בגרף קיים גם *click* וגם *Independent Set* (*IS* - קב' בת"ל) בגודל *k* כל אחד ?** הבעיה ב-***NPC***. עושים רדוקציה מ-*click* אחר. מוסיפים לגרף *k* קודקודים שאינם קשורים לכלום, ובהם בהכרח ה-*IS*. בגרף החדש יש *click* אם-ורק-אם היה במקור *click*.
2. ***בהינתן גרף, מספר k ושני קודקודים t ו-s – האם יש click בגודל kהמורכב משכני s והאם יש IS בגודל k של שכני t.*** הבעיה ב-***NPC***.
עושים רדוקציה מ-*click*. יוצרים צומת חדש *s'* ומחליפים בין *s* לבינו (כל הקשתות מ-*s* מנתקים ומעבירים ל-*s'*). עכשיו מחברים את *s* לכל הצמתים האחרים בגרף. בנוסף, מחברים ל-*t* *k* צמתים חדשים שמחוברים בקשת רק ל-*t*. מכאן כי בהכרח יש *IS* בגודל *k* שכנים של *t*, ויהיה *click* כנדרש אם-ורק-אם היה במקור *click* שהוא (כי כולם עכשיו שכנים של *s*, ובגדול הגרף המקורי לא שונה כלל).
3. **האם בגרף קיים *click* וגם *IS* בגודל *n/2+1* כ"א?** הבעיה ב-***P***. אם יש מספר זוגי של צמתים זה בלתי-אפשרי. אם יש מס' אי-זוגי זה אפשרי אבל אז זה בדיוק *n/2+1* בכל קב'. נספור כמה קשתות יש לכל צומת ואת **כל אלו עם יותר מ-*n/2+1* נשים בקבוצה אחת ואת השאר בשניה**. אם קיימים קליק ו-*IS* כנדרש, אז זו החלוקה ביניהם, ובדיקה האם בגרף בעל *d* צמתים יש קליק בגודל *d* היא קלה ופול" כי צריך לבדוק שבדיוק כולם מחוברים לכולם. כל התהליך פול" ולכן הכל פול". משהו כזה, צריך לשפצר מעט בהוכחה.
4. **האם בגרף יש צומת מדרגה *lg₂(n)* לפחות <u>או</u> קליק בגודל *m* ?** הבעיה ב-***P***. לבדוק דרגת צומת זה פשוט. אם אין, אז הקליק המקס' הוא פחות מ-*lg(n)*. לכן אם *m* גדול מזה – אין. אם *m* קטן מזה אז צריך לבדוק את כל האפשרויות. יש בסה"כ (lg_2n מעל *m*) אפשרויות, וזה יוצר *O(n)*. כל מס' פולי' של
$$\binom{\log_2 n}{m}=O(2^{\log_2 n})=O(n)$$
בדיקות וכל בדיקה היא בזמן פול" ולכן ב-*P*. נובע מכך ש:
5. **האם בגרף יש שני קליקים זרים בגודל *k*.** בעיה ב-***NPC***. רדוקציה מקליק. לוקחים את הגרף המקורי ומשכפלים אותו (ללא קשר בין שני החלקים). אם במקור היה קליק אז עכשיו יש שניים ואחרת לא.
6. **האם בגרף יש קליק בגודל *k* <u>או</u> קב' בת"ל בגודל *k*?** הביעה ב-***NPC***.
רדוקציה מקליק. מוסיפים לגרף עוד |*v*| צמתים ומקשרים אותם לכולם (כולל עצמם). עכשיו מחפשים בגרף החדש את הבעיה עבור *k+|v|*. בהכרח אין שם *IS* בגודל הזה כי קישרנו את הצמתים החדשים לכל הצמתים האחרים. לגבי קליק, אם היה במקור בגודל *k* אז עכשיו יהיה בגודל *k+|v|* כי הוספנו קליק |*v*| וחיברנו לכולם. זהו.
7. **האם בגרף דו"צ יש *IS* בגודל *k*** – הבעיה ב-***P***. לוקחים צד אחד, ומוסיפים לכמות את כל מה שבצד שני ולא מחובר לא. אם לא טוב, עושים כנ"ל רק עם הצד השני. אם גם עכשיו לא טוב אז אין, ואחרת מצאנו בדרך.
8. **באם לגרף יש תת-גרף מושרה דו"ל עם *k* קודקודים בכל צד**. בעיה ב-***NPC*** – רדוקציה מבעית *IS* לקב' בגודל *k*. מוסיפים קב' קודקודים *B* בגודל *k* ומחברים בקשתות לכל הקודקודים במקור. אם במקור היתה קב' בת"ל *A* בגודל *k* אז עכשיו היא דו"צ עם *B* (אין קשתות בין חברי *A* כי הוא *IS* ואין ב-*B* כי בבניה הגדרנו שאין). אם בגרף החדש יש גרף-מושרה דו"צ בגודל *k* אז לפחות צד אחד של הגרף הוא צמתים מהגרף המקורי, וזה אומר שהיה בו *IS* בגודל *k*.
9. **נתונה קב' *S* של אברים ו-*H* אוסף תתי-קבוצות לא ריקות. האם יש קב' *T* חלקית ל-*S* כל שכל תת-קבוצה ב-*H* מכילה לפחות אבר אחד ב-*T***. הבעיה ב-***NPC***. רדוקציה מבעית ***Vertex Cover***.
הרדוקציה לוקחת את הגרף ומגדירה את *S* להיות קבוצת הקודקודים וכל תת-קבוצה להיות קשת (ככה שהקבוצות ב-*H* הן כל הקשתות האפשריות). אם בגרף המקורי יש *vertex-cover* בגודל *k* אז זה בדיוק קב' *t* בגודל *k* כך שלכל תת-קב' ב-*H* יש אבר אחד ב-*t*. וצד שני – אם יש קב' *t* כזו, אז זה אומר שיש *vertex-cover*.
10. **כנ"ל עבור קב' *H* בגודל *n-5*.** הבעיה ב-***P***. בגלל הגודל סך כל תתי-הקב' האפשריות הוא פולינומי (*n* מעל *n-5* או מעל *5*). לכן אפשר לעבור על כל הקבוצות, לספור מופעים של כל אבר, ובכל פעם לקחת את האבר המקסימלי ולמחוק את הקבוצות שממנו הגיע. נפסיק כשיגמרו כל הקבוצות, וזה גודל הקב' *t* המינימלי. נשאר רק לבדוק האם *k* גדול-שווה מהמספר המינימלי הזה.

נוספים:

- $Np = Co\_np$ אמ"מ קיימת בעיה ב *NPC* שהיא גם *Co_np*
- *halt e NP_hard* אבל לא ל *NPC*.

<u>בעיות חשובות</u>

PATH – בהינתן גרף מכוון ושני קודקודים – האם יש מסלול בין שני הקודקודים (אפשר שלא חוזר באותו צומת פעמיים). *PATH={G,s,t | G has directed path from s to t}*. **בעיה זו שייכת ל-<u>*P*</u>**.

RELPRIME – שני מספרים יוגדרו *RELPRIME (relative primality)* אם המכנה המשותף המקסימלי שלהם הוא 1. כיוון שניתן לפתור בעיה זו בעזרת האלגוריתם של אוקלידס (*gcd*) שהוא פולינומי, הרי שהבעיה כולה נמצאת ב-***P***.

SUBSET-SUM – נתונה קבוצת מספרים ומספר יעד. האם יש שילוב של המספרים שחיבורם נותן את מספר היעד. הבעיה היא ***NP***.

<u>הבעיות הבאות הן בעיות *NP-Complete*</u>

HAMPATH = *{G,s,t | G has Hamilton path from s to t}*

כלומר האם בגרף יש מסלול בין *s* ל-*t* שגם עובר בכל הקודקודים וגם לא עובר באף קודקוד פעמיים.

HAMCIRCUIT = *{G | G has Hamilton circuit}*

בהינתן גרף מכוון רוצים לדעת האם יש מעגל שעובר בכל קודקודי הגרף וגם לא עובר באף קודקוד פעמיים.

<u>הערה</u>: *hamcircuit* מוגדר לגרף מכוון אבל הוכח בתרגול כי הוא נכון גם לגרפים לא מכוונים. בעקרון גם *hampath* לא מכוון הוא *NPC*, אבל זה <u>לא הוכח</u> בתרגול, ובעיקרון צריך להוכיח...

CLIQUE – האם בגרף יש תת-גרף קשיר לחלוטין (שבו כל הקודקודים מחוברים לכל שאר הקודקודים בתת-הגרף).

K-CLIQUE – לרוב זו תהיה השאלה, והכוונה להאם יש *clique* עם *k* קודקודים.

SAT – בהינתן ביטוי בוליאני, האם יש לו השמה מספקת.

3SAT – כנ"ל, כאשר הביטוי הבוליאני הוא מהצורה הבאה: מורכב מביטויים שקשורים ביניהם בקשר "וגם" ($\wedge$), כאשר בכל ביטוי יש 3 ליטרלים הקשורים ביניהם בקשר "או" ($\vee$).

INDEPENDET-SET – נתון גרף ושואלים האם יש תת-קבוצה של קודקודי הגרף, המורכבת מקודקודים ש<u>אין</u> ביניהן קשתות המקשרות ביניהן באופן ישיר (כלומר בין כל 2 קודקודים בתת-הקבוצה) <u>אין</u> קשת בגרף המקורי.

Vertx-Cover – נתון גרף ומספר *k* ונשאלת השאלה האם ניתן לכסות את הגרף ע"י *k* קודקודים. כיסוי הוא תת-קבוצה של הקודקודים כך שלכל קשת בגרף אחד הקודקודים (לפחות) של הקשת יהיה בתת-הקבוצה.

כלומר: $\forall(a,b)\in E:\ a\in V'\ or\ b\in V'$

K-coloring – האם ניתן לצבוע את קודקודי הגרף ב-*k* צבעים. צביעה חוקית היא כזו שלקודקוד יש צבע שונה מכל שכניו (הקודקודים הקשורים אליו בקשת) – במילים אחרות זה לכל קשת *(u,v)* אז ל-*u* ול-*v* יש צבע שונה.

דוגמא חשובה: למקרה שרוצים רדוקציות מ-***3SAT*** לבעיות אחרות שקשורות לנוסחאות *SAT* או *CNF* אז כדאי לבצע <u>העברות</u> כאלו: לוקחים נוסחה מהצורה $(y_1\vee y_2\vee y_3)$ ומעבירים למשהו מהצורה הבאה: $(y_1\vee y_2\vee z_1)\wedge(\overline{z_1}\vee y_3\vee b)$ (למשל רדוקציה מ-*3SAT* ל-*≠SAT* שבה בכל ביטוי מספק יש לכל היותר שני ליטרלים מאותו ערך ולשלישי ערך שונה).