מודלים חישוביים – סיכום למבחן

אוטומטים:

<u>שפות / מחרוזות / הגדרות בסיסיות:</u>

- <u>א"ב:</u> Σ הוא אוסף סופי של תווים, סימנים.
- <u>מחרוזת / מילה:</u> רצף סופי של אותיות מא"ב מסוים, כאשר מספר האותיות הוא אורכה
- <u>המחרוזת הריקה:</u> ε הינה מחרוזת מאורך אפס.
- <u>שפה:</u> אוסף של מילים מעל א"ב מסוים. L הינה שפה מקיימת $L \subset \Sigma^*$, כאשר Σ^* הינו אוסף כול המילים מעל הא"ב Σ
- <u>השפה הריקה:</u> ϕ שפה שלא מכילה אף מילה. זה **לא** שקול ל $L = \{\varepsilon\}$
- <u>שרשור:</u> חיבור שתי מילים אחת לקצה השנייה לקבלת מילה חדשה.

<u>אוטומט סופי דטרמניסטי DFA/ שפות רגולריות:</u> מכונה בעלת מספר סופי של מצבים. כאשר מתקבלת מחרוזת קלט האוטומוט קורא אותה אות אות מההתחלה לסוף כאשר עבור כול אות הוא עובר בין המצבים שלו לפי פונקציית המעברים שמגדירה אותו. לאחר קראית כול המחרוזת אם הריצה הסתיימה במצב מקבל אז האוטומט יקבל את מחרוזת הקלט, אחרת הוא ידחה.

- <u>הגדרה פורמלית:</u> $DFA = (Q, \Sigma, \delta, q_0, F)$ כאשר Q זו קבוצה סופית של מצבים, Σ זה האלפבית שמעליו פועל האוטומט, $\delta: Q\ X\ \Sigma \to Q$ פונקציה שמגדירה את המעברים בין המצבים בהתאם לקלט, q_0 המצב ההתחלתי ו F קבוצת המצבים המקבלים, כלומר $F \subset Q$.
- <u>מודל החישוב:</u> בהנתן אוטומוט $M = (Q, \Sigma, \delta, q_0, F)$ נאמר שהוא מקבל מילה $w = w_1, \dots, w_n$ אם קיימת סדרת מצבים $r_0, \dots, r_n$ כך ש $r_0 = q_0$ ו $r_n \in F$ וכמו כן $\delta(r_i, w_{i+1}) = r_{i+1}$. כלומר התחיל במצב ההתחלתי, סיים במצב מקבל, ובדרך עבר בצורה חוקית. ניתן להגדיר את δ לעבוד על מילים. $\delta(q, y\sigma) = \delta(\delta(q, y), \sigma)$
- <u>שפה של אוטומט:</u> עבור אוטומט M נסמן $L(M)$ להיות שפת כול המילים שהאוטומט מקבל. לכול אוטומט סופי דטרמיניסטי קיימת ויחידה השפה שהוא מקבל.
- <u>שפה רגולרית:</u> שפה L תקרא רגולרית אם קיים DFA M כך ש $L(M) = L$.
  - $\{w | \#_1(w) mod 2 = 1\}$
  - $\{0^n 1^m | n, m \in N\}$
  - כול שפה סופית.

<u>אוטומט סופי לא דטרמיניסטי NFA:</u> הרחבה לאוטומט סופי דטרמינסטי, כעת יכולים להיות מספר אפשרויות שונות למעבר בין מצבים עבור אותו הקלט, או שעבור קלט מסוים בכלל לא מוגדר המעבר, אפילו ניתן לבצע מעבר בין מצבים עבור המחזורת הריקה. נשים לב שכול DFA הוא בפרט NFA, כלומר זה רק אולי מוסיף כוח. אפשר לחשוב על זה כאילו המכונה בודקת את כול האפשרויות במקביל, בכול מקום שיש כמה אפשרויות היא מתפצלת לכמה נתיבי חישוב ובודקת את כול האפשרויות. אם אחת מהאפשרויות הסתיימה במצב מקבל אז המכונה תקבל.

- <u>הגדרה פורמלית:</u> $M = (Q, \Sigma, \delta, q_0, F)$ כאשר השינוים הם $\delta: Q\ X\ \Sigma \to P(Q)$ כלומר לכול מצב וקלט מסוים יש קבוצה של מצבים אפשריים לעבור אליה. ו $q_0 \subset Q$ כעת המצב ההתחלתי זה למעשה קבוצה של מצבים התחלתיים.
- <u>מודל חישובי:</u> נאמר ש M מקבל מילה $w = w_1 \dots w_n$ אם קיים רצף $r_0, \dots, r_n$ של מצבים כך ש $r_0 \in q_0$ וגם $r_n \in F$ וגם $r_{i+1} \in \delta(r_i, w_{i+1})$

- <u>שקילות לאוטומט דטרמיניסטי:</u> לכול שפה שקיים אוטומט NFA שמקבל אותה קיים גם DFA שמקבל אותה. הצורה זה פשוט להגדיר אוטומט דטרמיניסטי שהמצבים שלו יהיו כול תת הקבוצות של מצבים של הלא דטרמיניסטי, והמצבים המקבלים יהיו כול תת הקבוצות שיש בהם מצב אחד מקבל לפחות. ולכן שפה רגולרית אמ"מ קיים NFA שמקבל אותה.
- <u>מעברי אפסילון:</u> מאפשר לעבור בין שני מצבים מבלי לקרוא אות מהקלט. כעת נגדיר $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ ולכן $\delta = \Sigma_\varepsilon\ X\ Q \to Q$, כלומר מילות הקלט כעת הן מילים מהשפה שבמקומות אקראים יש להן אפסילון אם האוטומט יקבל מילה אם מסעות אפסילון נאמר שהוא מקבל את אותה המילה כאשר מוציאים ממנה את כול מסעות האפסילון. נשים לב שמעברי אפסילון לא מוסיפים כוח, כי ניתן להגדיר אוטומט שקול לחלוטין בלעדיהם, על ידי הגדרה מחדש של פונקצית המעברים כך ש $\delta(R,a) = \{q\ |q \in E(S) and\ s \in \delta(r,a) for\ r \in E(R)\}$ זה קבוצת כול המצבים הנגישים מ q על ידי מסע אפסילון. $E(q)$

<u>פעולות רגולריות</u>: תהיינה A,B שפות אז הפעולות הבאות תקרא רגולריות, ואם A,B רגולריות אז גם הפעלת הפעולה עליהן תניב שפה רגולרית.

- <u>איחוד:</u> $A \cup B = \{x: x \in A\ or\ x \in B\}$. ההוכחה היא פשוט להריץ "במקביל" כלומר לבנות אוטומט סופי דטרמיניסטי שרץ על המכפלה הקרטזית של המצבים שלהם, ויקבל אם אחד מהמצבים במכפלה של המצב הסופי הוא מצב מקבל באחד מהאוטומטים.
- <u>שרשור:</u> $A||B = \{xy : x \in A, y \in B\}$. הוכחה פשוטה על ידי NFA. נחבר את המצבים המקבלים של האוטומט של A עם מעברים אפסילון עם המצב ההתחלתי של האוטומט של B.
- <u>כוכב קליני:</u> $A^* = \{x_1 x_2 \dots x_n : n \geq 0\ and\ x_i \in A\}$. נשים לב שתמיד $\varepsilon \in A^*$. הוכחה פשוטה עם NFA. נייצר מצב התחלתי מקבל (עבור אפסילון) ואז נחבר אותו עם מעברי אפסילון למצב ההתחלתי האמיתי, ואת המצבים המקבלים נחבר במעבר אפסילון למצב הזה.
- <u>משלים:</u> $\bar{A} = \{x \in \Sigma^* : x \notin A\}$. ההוכחה היא פשוט על ידי אותו האוטומט רק שהופכים את המצבים המקבלים להיות המשלים שלהם, ולכן האוטומט החדש יקבל את המשלים, ולכן הוא רגולרי.

<u>GNFA אוטומט לא דטרמיניסטי מוכלל:</u> הכלל טבעית של NFA כך שיש רק מצב מקבל אחד ורק מצב התחלתי אחד, ויש חצים יוצאים מהמצב ההתחלתי לכול מצב אחר, ויש חצים נכנסים למצב המקבל מכול מצב אחר. בנוסף לכול מצב חוץ משני אלה יש חץ יוצא לכול מצב אחר. המעברים עתה מוגדרים לפי ביטויים רגולריים והמעבר בין שני מצבים יתבצע אם רצף האותיות הבא מהקלט תואם שייך לשפה של הביטוי הרגולרי שמגדיר את המעבר.

- <u>הגדרה פורמלית:</u> $M = (Q, \Sigma, \delta, q_s, q_a)$ כך שהשינוי הוא $\delta: (Q \setminus \{q_a\})\ X\ (Q \setminus \{q_s\}) \to R$ כלומר לכול זוג מצבים (מלבד הראשון והאחרון באופן שתואר) מתאים ביטוי רגולרי שמעביר בינהם.
- <u>מודל חישובי:</u> נאמר ש M מקבל מילה w אם קיימת חלוקה $w = w_1 \dots w_k$ כך שקיים רצף מצבים $q_0, \dots, q_k$ כך ש $q_0 = q_s$ ולכול i מתקיים $R_i = \delta(q_{i-1}, q_i)$: $w_i \in L(R_i)$ ו $q_k = q_a$
- <u>צמצום:</u> בהנתן אוטומט GNFA בעל K>2 מצבים ניתן לייצר אחד שקול בעל k-1 מצבים על ידי בחירת מצב כלשהו שאינו המקבל/ ההתחלתי, ועקיפתו על ידי הגדרה מחדש של הביטויים הרגולרים בין כול שאר המצבים. השקילות נובעת כי ההגדרה החדשה מאפשרת לעקוף את המופעים של המצב המדובר ברצף המצבים בריצה מקבלת של האוטומט.

<u>ביטוי רגולרי:</u> דרך נוחה ופשוטה להציג שפות רגולריות.

- <u>הגדרה פורמלית:</u> R הוא ביטוי רגולרי אם $R = a, a \in \Sigma$ או $R = \varepsilon, \phi$ או שהוא איחוד/ שרשור/ סגור קליני של ביטוים רגולריים אחרים. (הגדרה רקוריסיבית).

- <u>שפה של ביטוי רגולרי:</u> עבור ביטוי רגולרי R נגדיר את השפה שלו להיות או האות הבודדות שמרכיבה אותו, או המילה הריקה, או השפה הריקה, או איחוד/שירשור/כוכב של השפות של הביטויים הרגולריים שמרכיבים אותו.
- <u>שקילות לשפות רגולריות:</u> שפה רגולרית אמ"מ יש ביטוי רגולרי שהיא השפה שלו. ברור שבהנתן ביטוי רגולרי ניתן לבנות אוטומט שמקבל את השפה שלו כי ראינו איך בונים אוטומט לאיחוד /שרשור /כוכב , ואות בודדת זה רגולרי. בהנתן שפה רגולרית נסתכל על האוטומט שלה, נהפוך אותו ל GNFA על ידי הוספת מצב התחלתי חדש עם מעבר אפסילון, ומצב מקבל בודד עם מעברי אפסילון, ונתחיל לצמצם אותו עד אשר נשאר עם שני מצבים בלבד, ואז הביטוי שמקשר בינהם יהיה הביטוי הרגולרי שמייצג את השפה, מהשקילות על צמצום אוטומט.

<u>שפות לא רגולריות:</u> באופן אינטואיטיבי שפות שידרשו זיכרון לא חסום יהיו לא רגולריות, כלומר לא יהיה אפשר לקבלן עם אוטומט סופי דטרמיניסטי, אבל זה לא מספיק להוכחה.

- <u>למת הניפוח:</u> זה תנאי הכרחי(אבל לא מספיק!!) להיות שפה רגולרית, ולכן אם נקבל שפה שלא מקיימת את זה נדע שהיא לא רגולרית. הלמה אומרת שבהנתן L רגולרית קיים אורך קריטי l כך שלכול מילה $w \in L$ כך ש $|w| \geq l$ אז קיימת חלוקה $w = xyz$ כך ש $|y| > 0$ ולכול $i \geq 0$ מתקיים $xy^iz \in L$ בנוסף גם $|xy| \leq l$. נובע מיידית מכך שכמות המצבים סופית ולכן עבור מילה ארוכה מספיק חייבים לחזור על מצב פעמיים משובך היונים, ולכן על הקטע שהוביל מהמופע הראשון למופע השני של המצב הזה ניתן לחזור כמה פעמים שרוצים, או לא לבצע אותו בכלל. כדי להוכיח ששפה לא רגולרית נראה שלכול אורך l קיימת מילה בשפה כך שלכול חלוקה שלה קיים קבוע נפוח שהמילה המנופחת לא בשפה.
- <u>תכונות סגור:</u> ראינו ששפות רגולריות סגורות לאיחוד / שרשור / כוכב / משלים, ומכך נובע שגם לחיתוך, מחוקי דה מורגן. לכן אם יש לנו שפה שכשחותכים / מאחדים / משרשרים אותה עם שפה רגולרית מקבלים משהו לא רגולרי אז השפה לא רגולרית.
  - <u>פעולת החלוקה:</u> $L1/L2 = \{x: \exists y \in L2, xy \in L1\}$ נשים לב כי אם L_1 רגולרית אז L_1/L_2 לכול שפה L_2, תהיה רגולרית, ע"י הסתכלות על האוטומט שמקבל אותה ושינוי המצבים המקבלים כך שהם יהיו המצבים שמהם ניתן להגיע למצב מקבל בעזרת מילה ב L_2 נשים לב שקשה מאוד לחשב את זה, אבל קיים אוטומט כזה.
  - <u>הומומורפיזם:</u> פונקציה שממפה אותיות מא"ב Δ למילים מעל הא"ב Σ כלומר $h: \Delta \to \Sigma^*$, וניתן להגדיר גם את ההפעלה שלה על מילה שלמה, ואפילו על שפה באופן ברור. שפות רגולריות סגורות להומומורפזים, מיידי מהפעלת ההומומרופיזם על הביטוי הרגולרי.
  - <u>הומומורפיזם הפוך:</u> $h^{-1}(w) = \{x: h(x) = w\}$,$h^{-1}(L) = \{x: h(x) \in L\}$. נשים לב שמתקיים $h\left(h^{-1}(L)\right) \subset L \subset h^{-1}(h(L))$. שפות רגולריות סגורות להומומורפיזם הפוך. הוכחה על ידי הגדרת אוטומט חדש שהמעברים בו יגדרו על פי המעברים של האוטומט המקורי אחרי שהפעלנו את h עליהם.
- <u>מהיל נירוד ומחלקות שקילות:</u>
  - בהנתן שפה $L \subset \Sigma^*$ היא מגדירה יחס שקילות על כול המילים ב Σ^* באופן הבא: עבור $x, y \in \Sigma^*$ נאמר ש $x \sim_L y$ אם לכול מילה $z \in \Sigma^*$ מתקיים ש $xz \in L \leftrightarrow yz \in L$ כלומר הם מתנהגות באותו אופן עבור כול המשכה שלהן.
  - יחס השקילות לעיל מחלק את כול המילים בעולם למחלקות שקילות, אנחנו נתעניין כמה כאלה יש, וזה יתן אינפורמציה על השפה. (כדי להוכיח שלשפה יש אינסוף מחלקות שקילות צריך למצוא סדרה אינסופית של מילים שכול אחת במחלקת שקילות אחרת)
  - <u>משפט MN:</u> שפה רגולרית אמ"מ מספר מחלקות השקילות סופי.
    - $\leftarrow$ אם השפה רגולרית אז יש אוטומט שמקבל אותה, ולו יש כמות סופית של מצבים. נגדיר יחס שקילות על מילים להיות מילים שהגיעו לאותו המצב לאחר שהאוטומט רץ עליהם, ברור שזה יחס שקילות עם כמות מחלקות סופית. ברור כי

אם שתי מילים שקולות לפי היחס הזה הן גם שקולות לפי היחס שהשפה משרה, ולכן גם ליחס השפה חייב להיות כמות סופית של מחלקות.

- ▪ $\rightarrow$ אם יש מספר סופי של מחלקות שקילות נגדיר אוטומט שהמצבים שלו הם נציגים של מחלקות השקילות, והמעברים בינהם הם לפי הדבקה מימין, והמצבים המקבלים הם כאלה שהנציג שלהם בשפה. לכן האוטומט הזה מקבל את השפה, ולכן רגולרית.
  - אוטומט מינמלי: מספר מחלקות השקילות ביחס השקילות של האוטומט הוא מספר המצבים, וראינו שמספר מחלקות השקילות של השפה קטן שווה למספר מחלקות השקילות של האוטומט (כי נניח שהיו יותר מחלקות ביחס השפה, אז בהכרח שתי מילים ששקולות לפי האוטומט לא היו שקולות לפי השפה, סתירה) ולכן יש חסם תחתון למספר המצבים המינמלי של האוטומט, כמו כן עכשיו נראה שהחסם הדוק, כלומר קיים אוטומט מינמלי לשפה.
  - אלגוריתם לקבלת המינימלי: נתחיל משתי קבוצות מצבים, קבוצת המצבים המקבלים וקבוצת כול השאר. כול עוד יש קבוצה כך שיש בה שני מצבים שהם לא באותה מחלקת שקילות אז נפצל הקבוצה לשתיים לפי האות שמראה שהמצבים לא שקולים, ונמשיך ברקורסיה. מובטח שנקבל שמספר הקבוצות הוא כמספר מחלקות השקילות של השפה, ואלה יהיו המצבים של האוטומט המינמלי.

- דוגמאות לשפות לא רגולריות:
  - $\{0^n 1^n | n \in N\}$, הוכחה מלמת הניפוח.
  - $\{\#_1(w) = \#_0(w)\}$ הוכחה מלמת הניפוח.
  - $\{0^i 1^j : i > j\}$ הוכחה מלמת הניפוח, וניפוח בקבוע 0.
  - $primes = \{1^p : p\ is\ prime\}$ על ידי למת הניפוח.

בדיקת קשרים / תכונות של אוטומטים סופיים: יהי NFA נסמנו N, נרצה לראות אילו תכונות אנחנו יכולים לדעת עליו.

- האם $w \in N$? נהפוך את N ל DFA ונריץ. (יכול לקחת הרבה זמן)
- האם $L(N) = \phi$? זו למעשה שאלה על קשירות בגרף, האם יש מסלול כלשהו מצומת אחד לצומת אחר. ניתן לפתור דיי מהר.
- האם $L(N) = \Sigma^*$? שקול ללבדוק האם המשלים ריק, ולמצוא משלים זה קל (אפשר למשל לבנות אוטומט דטרמניסטי ולהפוך המצבים המקבלי)
- האם $L(N_1) \subset L(N_2)$? זה כמו לבדוק האם החיתוך עם המשלים ריק. ראינו איך בונים אוטומט לחיתוך.
- האם $L(N_1) = L(N_2)$? שקול ללבדוק הכלה דו כיוונית.

דקדוק חסר הקשר: אוסף של משתנים וכללי גזירה שמגדירים בהנתן משתנה איך לגזור ממנו משתנים חדשים + ליטרלים בשפה. יש משתנה התחלתי, וכול הרצפים של ליטרלים בלבד, שניתן לגזור מהמשתנה ההתחלתי, יהיו מילים בדקדוק. נשים לב שתהליך הגזירה לא דטרמניסטי כי יש כמה דרכים שונות לגזור כול משתה, אחרת תמיד הינו מקבלים מילה בודדת. נשים לב אולם שלסדר בו אני מבצע את כללי הגזירה אין חשיבות לתוצאה הסופית. אוסף המילים שניתן לגזור מדקדוק יקראו השפה של הדקדוק, וכול שפה שקיים דקדוק שהיא השפה שלו תקרא שפה חסרת הקשר.

- הגדרה פורמלית: $G = (V, \Sigma, R, S)$ כאשר V זה אוסף סופי של משתנים, Σ זה אוסף סופי של ליטרלים (הא"ב), R אוסף סופי של כללי גזירה, כאשר כלל גזירה זה מתשנה ומחרוזת סופית של משתנים וליטרלים, S זה המשתנה ההתחלתי. נאמר ש $u \rightarrow v$ אם קיים כלל גזירה בדקדוק כך שהוא לוקח משתנה מתוך u ולאחר ביצוע הגזירה מתקבל v. נאמר ש $u \rightarrow^* v$ אם קיים רצף של גזירות שיובל מ u ל v (או שהם שווים). נסמן $L(G) = \{w : S \rightarrow^* w\}$ להיות השפה של הדקדוק חסר ההקשר.

- עץ הגזירה: בהנתן גזירה של מילה מסוימת בדקדוק ניתן לבנות עץ שמייצג את תהליך הגזירה כאשר השורש הוא המשתנה ההתחלתי, וכול התפצלות מייצגת את התוצאות של כלל הגזירה שבחרנו באותו השלב, כאשר הבנים של הצומת הם המשתנים והליטרלים שנוצרו מכלל הגזירה.
- כדי להוכיח ששפה כלשהי היא חסרת הקשר נצטרך להראות הכלה דו כיוונית, כלומר צריך להראות דקדוק שכול מילה שהוא גוזר היא בשפה, ושאין מילה בשפה שאי אפשר לגזור אותה מהדקדוק. בדרך כלל נעשה זאת באנדוקציה שתטען טענה חזקה בהרבה ממה שאנו רוצים להוכיח.
- נשים לב שיש דקדוקים שהם דו משמעיים, כלומר את אותה המילה ניתן לגזור בשני עצי גזירה שונים לחלוטין! וזה יכול להוות בעיה לעיתים, כי לפעמים נרצה להיות מסוגלים לשחזר את עץ הגיזרה מהמילה הסופית. לא לכול דקדוק דו משמעי יש דקדוק חד משמעי ששקול לו! אבל לחלק מהן כן יש.
- CNF: Chomsky normal form: זו צורה קנונית לייצוג דקדוק חסר הקשר, כך שבצורה הזו כללי הגזירה הם אך ורק ממשתנה לליטרל, או ממשתנה לשני משתנים(שאף אחד מהם הוא לא ההתחלתי), בנוסף המשתנה היחיד שיכול לגזור את אפסילון, אם בכלל זה המשתנה הראשון. נשים לב שכול דקדוק חסר הקשר אפשר להפוך לצורה הזו.
- בדיקת שייכות לשפה חסרת הקשר: כלומר בהנתן דקדוק ומילה האם הדקדוק הזה גוזר את המילה הזו? תחיל נעביר הדקדוק לצורת CNF. לכול מילה $w \in L(G)$ כאשר G בצורה CNF $|w| = n$ אז מספר צעדי הגזירה הוא 2n-1 (מוכיחים על ידי הסתכלות על עץ הגזירה שכול צומת בו מייצג צעד גזירה, ויש לו n עלים). לכן ניתן להגדיר פונקציה שתקבל מילה ומשתנה ותחזיר האם המשתנה גוזר אותה על ידי שלכול כלל שהופך את המשתנה לשני משתנים ולכול חלוקה של המילה לשני חלקים נמשיך ברקורסיה ואם אורך המילה הוא אחד פשוט נבדוק אם המשתנה גוזר הליטרל הזה, מובטח שהאלגוריתם יעצור כי כול פעם מקטינים את אורך המילה בלפחות אחד. אם משתמשים בתכנות דינמי אפילו אפשר לפתור הבעיה בזמן פולינומיאלי!
- שפות רגולריות הן חסרות הקשר: בהנתן DFA נגדיר דקדוק שעוקב אחרי מעברי האוטומט, כלומר המשתנים שלו יהיו המצבים, וכללי הגיזרה יהיו ממשתנה אחד לליטרל כלשהו ומימין לו המשתנה הבא, כאשר הליטרל הזה מעביר באוטומט מהמצב הראשון לשני.
- פעולת ה reverse: נשים לב שאם L חסרת הקשר אז גם $R(L) = \{w^R : w \in L\}$ גם חסרת הקשר, הוכחה באינדוקציה, והחלפת כללי הגזירה לגזור את אותן המילים רק ברוורס.
- תכונות סגור של שפות חסרות הקשר:
  - איחוד: נגדיר את כלל הגזירה הראשון הבא $S \to S_1|S_2$
  - שרשור: נגדיר את כלל הגזירה הבא $S \to S_1S_2$
  - כוכב: $S_{new} \to \varepsilon|S_{old}|S_{new}S_{new}$ (כלומר מספר המילים שיהיו נקבע כבר בהתחלה, ואז פושט גוזרים כרגיל.
  - **לא** סגורות לחיתוך: נשים לב שלא ניתן להריץ במקביל אוטומט מחסנית על שתיהן כמו שעשינו עבור DFA כי הזכרון יצטרך להיות משותף וזה בעיה, כי יש רק מחסנית אחת וריצה על שפה אחת תדפוק את הריצה על השפה השניה. ולכן חיתוך של שפה חסרת הקשר עם שפה רגולרית **כן** יהיה חסר הקשר, כי השפה הרגולרית לא "תבזבז" מקום בזכרון...
  - **לא** סגורות למשלים: נובע מיידית מהסגירות לאיחוד אבל אי הסגירות לחיתוך וחוקי דה מורגן.
  - הומומורפיזים: ברור ששפות חסרות הקשר סגורות להומומורפזים, פשוט בדקדוק כול פעם שרצינו לגזור ליטרל מסוים, במקום זה נגזור את הליטרל שהוא ההומומורפיזם מופעל עליו.
  - הומומורפיזם הפוך: הרעיון הוא פשוט להריץ את האוטומט של המילה אחרי שהפעלנו עליה את ההומומורפיזם, הבעיה היא שלא יהיה לנו איפה לשמור את המילה, ולכן נשתמש ב buffer, כלומר נחשב את המילה בעת הצורך, ומשום שיש חסם על האורך המקסימלי

של מילה אחרי שהפעלנו עליה את ההומומורפיזם ניתן לשמור את זה במצבים עצמם ולא לבזבז מקום במחסנית.

- <u>דוגמאות לשפות חסרות הקשר:</u>
  - $\{a^n b^n : n \in N\}$
  - $\{ww^R : w \in \Sigma^*\}$
  - $\{\#_1(w) = \#_0(w)\}$

<u>שפות שאינן חסרות הקשר:</u>

- <u>למת הניפוח:</u> L שפה חסרת הקשר אז קיים אורך מינמלי l כך שלכול מילה $w \in L$ כך שמתקיים $|w| \geq l$ אז קיימת חלוקה $w = uvxyz$ כך שלכול $i \geq$ מתקיים $uv^i xy^i z \in L$ וכמו כן $|vy| > 0$ וגם $|vxy| \leq l$ (כלומר זה לא טריוויאלי ולא יכול לקרות רחוק מדיי מההתחלה). ההוכחה היא שיש כמות סופית של משתנים, ולכן עבור מילה שארוכה מ$2^{|V|+1}$ (או אם לא מדובר ב CNF אז זה יהיה גדול מ 2 אבל עדיין סופי) מובטח שיש מסלול בעץ שאורכו לפחות $|V| + 1$ ולכן משובך היונים יש משתנה שחוזר על עצמו... (שוב זה תנאי הכרחי להיות שפה חסרת הקשר אבל זה לא תנאי מספיק)
- <u>דוגמאות לשפות שאינן חסרות הקשר:</u>
  - $\{a^n b^n c^n : n \in N\}$ מלמת הניפוח.
  - $\{ww : w \in \{0,1\}^*\}$ (המילה המנצחת היא $0^l 1^l 0^l 1^l$

<u>אוטומט מחסנית :PDA</u> מיועד להיות מקבל עבור שפה הנוצרת מדקדוק חסר הקשר. באופן לא פורמלי זה אוטומט מצבים בעל מחסנית אינסופית שמסוגל להכניס דברים למחסנית ולהוציא דברים ממנה ולשנות את אופן הפעולה שלו לפי מה שיש במחסנית, מגלם בתוכו הרבה אי דטרמיניזם, וזה **לא** שקול לאוטומט מחסנית כן דטרמיניסטי.

- <u>הגדרה פורמלית:</u> $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ כאשר Q קבוצה סופית של מצבים, Σ אוסף סופי של אותיות שמהם יורכב הקלט, Γ אוסף סופי של אותיות שבהם המחסנית יודעת לטפל, פונקציות מעברים $\delta: Q\ x\ \Sigma_\varepsilon\ x\ \Gamma_\varepsilon \to \mathrm{P}(\mathrm{Q}\ x\ \Gamma_\varepsilon)$ (כלומר בהנתן מצב נוכחי אות של קלט ואת האות בראש המחסנית תחזיר את קבוצת זוגות המצבים וראשי מחסנית חדשים האפשרית לעבור מכאן, כזכור או דטרמיניזם), $q_0 \in Q$ מצב התחלתי ו $F \subset Q$ מצבים מקבלים.
- <u>מודל חישובי:</u> עבור מילה $w = w_1 \dots w_n$ נסמן $\delta^*(w)$ להיות אוסף כול הזוגות (q, s) כאשר q מצב ו s מחרוזת של תווים מהא"ב של המחסנית, כך שלאחר ריצה של האוטומט על w הוא יכול להגיע למצב q כאשר במחסנית רשומה המחרוזת s. נאמר שהאוטומט מקבל מילה אם מצב מקבל כך שעבור איזה שהיא מחרוזת t ועבור איזשהי דרך של שילוב אפסילונים בתוך המילה w יתקיים ש $\delta^*(w')$ מכיל את המצב המקבל.
- <u>שפות רגולריות ואוטומטי מחסנית:</u> מיידי לראות ששפה רגולרית בפרט קיים אוטומט מחסנית שמקבל אותה, כי אוטומט רגיל הוא גם אוטומט מחסנית שלא מנצל את המחסנית שלו. לכן $L_{DFA} \subsetneq L_{PDA}$
- <u>שקילות לשפות חסרות הקשר:</u> שפה היא חסרת הקשר אמ"מ קיים אוטומט מחסנית שמקבל אותה.
  - ← בהנתן דקדוק חסר הקשר נבנה אוטומט מחסנית שבהתחלה יכניס את המשתנה הראשון למחסנית ואחריו סימן דולר, ואז כול פעם יגזור את המשתנה השמאלי ביותר לפי כלל גזירה כלשהו (באופן לא דטרמיניסטי הוא יבחר אחד) ויכניס את התוצאה למחסנית. כול עוד יש ליטרלים משמאל למשתנה הכי שמאלי שהם לא משתנים הוא ישווה אותם עם אותיות מהקלט, ואם אין שוויון מסלול החישוב הזה ימות. אם הוא הגיע לסימן הדולר סימן שהוא השווה את כול המילה ולכן הוא יקבל.
  - → בהנתן אוטומט מחסנית נגדיר דקדוק חסר הקשר. המשתנים ייצגו מעברים בין מצבים כך שלפני המעבר המחסנית הייתה ריקה ואחרי היא גם ריקה, והמשתנה הראשון יהיו

משתנה שמייצג מעבר מהמצב הראשון למצב מקבל (יחיד) כשהמחסנית ריקה בסוף.
נוסיף כללי גזירה שיאפשרו לעבור במצבים באמצע כלומר $A_{pq} \to A_{pr}A_{rq}$, נוסיף $A_{qq} \to \varepsilon$ וגם אם אפשר להתחיל ממצב מסוים תוך קריאת קלט מסוים, לכתוב משהו במחסנית, ואז במצב אחר לקרוא קלט נוסף ולמחוק את הדבר הזה שהיא במחסנית אז נוסיף את המעבר הזה תוך כתיבת הליטרלים שנקראו מהקלט.

<u>בדיקת קשרים / תכונות של שפות חסרות הקשר:</u>

- האם $L(G) = \phi$? כלומר האם הדקדוק גוזר איזשהי מילה? אפשרי בזמן סופי! תחילה נסמן את כול הטרמינלים בדקדוק, עכשיו בלולאה נסמן כול פעם משתנה שכלל הגזירה שלו גוזר דברים שכולם כבר מסומנים, ברגע שאיטרציה מסוימת לא סימנו אף משתנה אז נעצור ונמחק את כול המשתנים הלא מסומנים. אם מחקנו את המשתנה ההתחלתי אז השפה ריקה, אחרת יש מילה שניתן לגזור.
- האם $L(G) = \Sigma^*$ לא ניתן לפתור! כלומר זה לא בR . (נזכור שאין סגירות למשלים ולכן אי אפשר לבדוק אם המשלים ריק) (בפרט **אי אפשר להשוות בין שני דקדוקים**, כי יש רדוקציה לבעיה הזו)
- האם $|L(G)| < \infty$? אפשר לדעת! נעלים משתנים מיותרים כמו שעשינו בבדיקת ריקות, ואז נצייר את הגרף שמייצג את הקשרים בין המשתנים, אם יש מעגל אז השפה אינסופית.

<u>חישוביות:</u>

<u>מכונת טיורינג:</u> למכונה יש סרט זכרון אינסופי לימין, שעליו רשום הקלט. למכונה יש ראש קריאה / כתיבה ובכול שלב היא יכולה לבחור מה לכתוב בתא שהרגע קראה, ולבחור האם לזוז שמאלה או ימינה, כלומר היא יכולה לקרוא את הקלט כמה פעמים שהיא רוצה, והיא יכולה לכתוב זיכרון שתוכל לגשת אליו בעתיד, בנוסף למכונה יש כמות סופית של מצבים שמגדירים לה איך לבצע את הקריאות / כתיבות, וגם יש לה מצב מקבל / דוחה והיא בוחרת מתי לעבור למצבים האלה, אין מושג של סוף הקלט.

- <u>הגדרה פורמלית:</u> $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ כאשר Q זו קבוצה סופית של מצבים, Σ זה האלפבית של הקלט, Γ זה האלפבית של הסרט, כלומר מה המכונה יכולה לכתוב, זה יכיל את Σ ועוד אותיות, $\delta: Q\ x\ \Gamma \to Q\ x\ \Gamma\ x\ \{R, L\}$ פונקציית המעברים שבכול שלב יודעת מה המצב הנוכחי ומה כתוב על הסרט, ואומרת לאיזה מצב לעבור, מה לכתוב על הסרט במקום מה שרשום, והאם לזוז שמאלה או ימינה. q_0 המצב ההתחלתי, q_a, q_r המצבים המקבל והדוחה.
- <u>מודל חישובי:</u> הסרט מתחיל כך שהקלט רשום ברצף מהקצה השמאלי, וכול שאר התאים בזכרון מכילים תו _ שאינו שייך Σ ולכן מסמנים את סוף הקלט. המכונה רצה לפי פונקצית המעברים עד אשר הגיעה למצב $q_a\ or\ q_r$, כלומר היא יכולה לרוץ לנצח. (אם המכונה מנסה לזוז שמאלה מהקצה הכי שמאלי אז הראש לא זז..)
- <u>קונפיגורציה:</u> מחרוזת סופית אשר מייצגת את מצבה של מכונת הטיורינג ברגע מסוים.
הקונפיגרוציה בנויה מהתווים שמופיעים על הסרט כרגע (יש רק כמות סופית בכול רגע נתון) ואת מיקום הראש, שייצון על ידי שם המצב שבו נמצאת המכונה כרגע, והוא יופיע משמאל לאות שעליה כרגע הראש נמצא. נגדיר קונפיגורציה עוצרת להיות כזאת שמכילה את המצב המקבל / המצב הדוחה.
- <u>קבלת מילה:</u> נאמר שמכונת טיורינג M מקבלת מילה w אם קיים רצף של קונפיגרציות כך שהראשונה היא ההתחלתית, האחרונה היא מקבלת ובדרך כול המעברים היו חוקיים. אוסף המילים שהמכונה מקבלת יקרא השפה שלה.

- <u>מודלים שקולים:</u>
  - לאפשר לסרט להשאר במקום: לא מוסיף כוח כי ניתן פשוט לעשות שני מצבים פיקטיבים של לזוז ימינה ואז חזרה שמאלה. אפשר לסמלץ כול אחד מהמודלים בעזרת השני ולכן הם שקולים.
  - הרבה סרטים: מכונה שיש לה כמות סופית של סרטים ובכול שלב המכונה קוראת את מה שכתוב בכול הסרטים יחד, כותבות ביחד בכול הסרטים (כול סרט משהו אחר) וזזה בכול סרט לאן שהיא רוצה (ימינה או שמאלה בלי קשר לאן הראש זז בסרטים אחרים). ברור שמכונה מרובת סרטים יכולה לסמלץ מכונה רגילה, נראה את הכיוון השני. פשוט נשמור את הזכרון של כול הסרטים בסרט אחד מופרדים על ידי סימן מזהה. בנוסף נגדיל את האלפבית שלנו להכיל בנוסף את אותם אותיות אבל עם סימן מזהה שהראש מצביע עליהן כרגע, והריצה תהיה פשוט לקרוא את כול הסרט ומכול תת קטע לקרוא איזה אות הראש מצביע עליה עכשיו, לחזור להתחלה ואז בהתאם למה שקראנו לרוץ על כול תת הקטעים ובכול תת קטע לשנות את הסרט ואת מיקום הראש בהתאם. יש רק כמות סופית של סרטים ולכן את כול הזכרון הנוסף הזה אפשר לשמור במצבים של המכונה.
  - מחשב עם זכרון RAM ותכנות בשפת מכונה. אפשר בקלות לסמלץ על מכונת טיורינג מרובת סרטים. בגדול כול מודל של מחשב ניתן לסמלץ על מכונת טיורינג (כרגע לא אכפת לנו שזה יקח המון זמן, העיקר שאפשר)

<u>Turing complete</u>: מודל חישוב יקרא טיורינג קומפליט אם הוא מסוגל לסמלץ מכונת טיורינג. Church-turing thesis אומרת שמכונת טיורינג יכולה לסמלץ כול מודל חישובי הגיוני. כלומר ההגדרה של אלגוריתם ומה אפשר ואי אפשר לחשב לא תלויה במודל החישובי עצמו. אותנו לא יענינו מכונת טיורינג בפני עצמן, אלא רק העובדה שהן מייצגות מה אפשר ומה אי אפשר לחשב.

<u>מכונות טיורינג לא דטרמיניסטית:</u> חישוב של מכונה לא דטרמינסטית הוא כמו עץ של מעברים בין קונפיגרציות כאשר השורש הוא ההתחלתית. נאמר שהמכונה מקבלת מילה אם אחד מהענפים של העץ מסתיים במצב מקבל, והיא דוחה מילה אם כול הענפים דוחים או אינסופיים.

- <u>פונקצית המעברים:</u> $\delta: Q\ x\ \Gamma \to P(Q\ x\ \Gamma\ x\ \{L,R\})$ כלומר מכול קונפיגורציה אפשר לעבור לכמה קונפיגורציות אפשריות באופן חוקי.
- <u>שקילות למכונה דטרמיניסטית מבחינת השפות שהיא יכולה לקבל:</u> נסמלץ מכונה לא דטרמיניסטית על ידי אחת כן דטרמינסטית שיהיו לה שלושה סרטים. אחד יהיה סרט הקלט שאותו לא נשנה, השני יהיה סרט העבודה, והשלישי יהיה סרט שיציין את הבחירות הלא דטרמיניסטיות שהתבצעו. כלומר הרעיון הוא שהמכונה תנסה את כול הענפים האפשריים בעץ ותקבל אם אחד מקבל, הבעיה היא שאי אפשר לנסות אותם לעומק קודם כי אנחנו יכולים להתקע באחד אינסופי, לכן נרצה לרדת בעץ לרוחב, ולכן כול שלב נחשב ענף מסוים מההתחלה (הקלט) עד לעומק מסוים לפי הבחירות הלא דטרמיניסטיות שנתונות בסרט הבחירות, אחרי שהגענו לעומק הרצוי (אם לא דחינו / קיבלנו בדרך) נעצור הריצה, נשנה את הסרט שבורר את הבחירות, ונחשב שוב הכול מההתחלה. מובטח לנו שנעבור על כול האפשרויות לבחירות לא דטרמנסטיות. (כמובן שזה יקח המון המון זמן...)

<u>המחלקה $R\varepsilon$:</u> כול השפות שקיימת מכונת טיורינג שמקבלת אותן (לא בהכרח תמיד עוצרת עבור קלטים שאינם בשפה!) (או באופן שקול קיימת מכונה מרובת סרטים / לא דטרמניסטית וכו'...)

- <u>אינומרטור:</u> מכונת טיורינג שפולטת את כול המילים של שפה L, ושום דבר אחר, תקרא אינומרטור עבור השפה L. כאשר לפלוט משמהו לרוץ ללא הפסקה ולרשום על הסרט את המילים אחת אחרי השנייה. למכונה אין קלט.
- <u>$L \in R\varepsilon \leftrightarrow \exists\ enumerator\ for\ L$</u> : אם קיים אינומרטור ניתן להגדיר מכונת טיורינג שבהתן מילה תסמלץ את האינומרטור ועבור כול מילה שהוא פולט תבצע השוואה ותקבל אם יש

שוויון עבור מילה כלשהי. המכונה הזו תקבל את השפה. (אבל לא תכריע!!). במידה ויש מכונת טיורינג שמקבלת את השפה נגדיר אינומרטור כמכונת טיורינג שעוברת על כול המילים בעולם בסדר לקסיקוגרפי ובכול איטרציה מאפשרת למכונה של השפה לרוץ כמות גדולה יותר של צעדים על המילים, אם המכונה קיבלה מילה כלשהי האינומרטור ידפיס אותה.

- סגירות: לאיחוד / חיתוך וכוכב. סגירות לכוכב אפשר למשל על ידי בניית מכונה לא דטרמינסטית שבוחרת איך לחלק את הקלט לתתי מילים ומריצה את המכונה המקורית על כול תת חלק(בשיטת הריצה המדוגרת) ומקבלת אמ"מ כול תת חלק קיבל.
- מוודא: שפה היא ב $R\varepsilon$ אמ"מ קיים לה מוודא (לא בהכרח פולינומיאלי!) העד יהיה פשוט מספר צעדי הריצה.

המחלקה R: נאמר שמכונת טיורינג מכריעה שפה אם היא מקבלת אותה, ודוחה כול מילה שאינה בשפה. במילים אחרות היא תמיד עוצרת ומקבל את השפה. שפה L תקרא כריעה אם קיימת מכונת טיורינג שמכריעה אותה ונסמן $L \in R$.

- אינומרטור מונוטוני: אינומרטור שמדפיס את כול המילים בשפה אבל בסדר לקסיקוגרפי.
- שפה כריעה אמ"מ יש לה אינומרטור מונוטוני: אם היא כריעה פשוט נריץ את המכונה על המילים לפי הסדר ונדפיס רק את אלה שהיא מקבלת. אם יש לה אינומרטור מונוטוני אז בהנתן מילה פשוט נריץ האינומרטור עד שהוא יעבור את המילה בסדר הלקסיקוגרפי, אם היא לא הופיעה סימן שהיא לא בשפה, ומובטח שזה יקרה בזמן סופי.
- סגירות: המחלקה סגורה למשלים כי פשוט נבנה מכונה שתחזיר הפוך, מובטח שגם היא תכריע.

המחלקה $Co - R\varepsilon$: כול השפות שהמשלים שלהן הוא ב $R\varepsilon$.

- $E_{TM} = \{< M > | L(M) = \phi\}$ (הטריק הידוע של להריץ כול פעם צעד אחד יותר)

$R = R\varepsilon \cap Co - R\varepsilon$: בברור R מוכל בחיתוך כי $L \in R \to L \in R\varepsilon$ כי מכונה שמכריעה בפרט גם מקבלת. ואם $L \in R$ אז על ידי הפיכת המצב המקבל והדוחה נקבל ש $\bar{L} \in R$ ולכן $\bar{L} \in R\varepsilon$ ולכן L בחיתוך. בכיוון השני אם השפה בחיתוך אז יש מכונה שמקבלת אותה ומכונה שמקבלת את המשלים לכן בהנתן מילה נריץ צעד כאן צעד כאן, מובטח שאחרי זמן סופי לפחות אחת מהשתיים תסיים לרוץ ותקבל / תדחה וככה נוכל להכריע את השפה. (להריץ במקביל משמהו מכונה עם שני סרטים שעושה בכול סרט את עבודת מכונה אחרת)

קידוד מכונת טיורינג / מכונה אוניברסלית: נשים לב שקלט למכונת טיורינג זה רצף של תווים, אבל לעיתים נרצה שהאלגוריתם שלנו ירוץ על כול מיני יצורים כמו גרפים, מטריצות וכו', לכן לכול ייצור יהיה לנו קידוד. עבור יצור x נסמן הקידוד שלו $< x >$. גם מכונת טיורינג עצמה יכולה להיות מקודדת כמחרוזת, וזה יאפשר לנו בהנתן מחרוזת תחילה לבדוק שהיא אכן מייצגת מכונת טיורינג חוקית, ואם כן, נוכל לסמלץ אותה על ידי מכונה אחרת, מכונה אוניברסלית שמסוגלת לקרוא קידודים של מכונת ולסמלץ אותן.

- קידוד סטנדרטי: נקודד מכונה לפי פונקציית המעברים שלה. נקודד את מספר המצב אליו עוברים בעזרת רצף של אפסים באורך המתאים, את מספר האות לכתוב גם בעזרת רצף אפסים, ואז הכיוון לזוז כנ"ל. הרצפים מופרדים על ידי אחדות, ומעברים שונים מופרדים על ידי צמד אחדות.
- מכונה אוניברסלית: מכונה בעלת כמות יחסית מצומצת של מצבים שמסוגלת לסמלץ מכונות בעלות כמות עצומה של מצבים. ממש כמו מפרש של פייטון לדוגמא.

בעיית העצירה/ בעיית הקבלה, שפות שלא ניתן להכריע / שפות שלא ניתן למנות:

- $A_{TM} = \{< M, w > | M \text{ is a turing machine that accepts } w\}$ הבעיה הזו מקיימת $A_{TM} \notin R$. נזכר שראינו שהבעיה הדומה עבור DFA/NFA/PDA כולן כריעות. נשים לב תחילה ש $A_{TM} \in R\varepsilon$ כי המכונה האוניברסלית תקבל את השפה הזו, כלומר המכונה האוניברסלית בהנתן קלט שהוא מכונה ומילה תקבל אם המכונה מקבלת את המילה.
- $\mathbf{A_{TM} \notin R}$: נניח בשלילה שקיים מכריע H לשפה A_{TM} כך שלכול קלט $< M, w >$ הוא מקבל אם M מקבלת את w ודוחה אחרת. נגדיר מכונה חדשה D שתקבל כקלט קידוד <M> של מכונה ותפעיל את H<M,M> כלומר האם M מקבל את המילה <M> , אם התשובה חיובית אז D תדחה, ואם התשובה שלילית D תקבל. כעת כשנריץ D(<D>) נקבל סתירה כי D תשאל את H האם D מקבל את <D> ומה ש H תחזיר D תתנהג אחרת... לכן H לא קיימת.
- $L \notin R$ אז בהכרח $L \notin R\varepsilon \; or \; \bar{L} \notin R\varepsilon$ כי אחרת L הייתה בחיתוך שהוא כול השפות שניתן להכריע, אבל אי אפשר להכריע אותה...
- $\overline{A_{TM}} \notin R\varepsilon$ מהטענה לעיל. כלומר כול המכונות ומילים שהמכונה לא מקבלת. אינטואטיבית בשונה מA_{TM} שניתן להריץ עד אשר המכונה תקבל ולכן זה ב $R\varepsilon$ את השפה הזו לעולם לא נוכל להחליט מתי להפסיק להריץ, כי מרבית המילים בשפה הן מילים שגורמות למכונה לרוץ עד אינסוף, אז לעולם לא נדע מתי להפסיק ולהגיד שמכאן המכונה לעולם לא תקבל.
- $H_{TM} = \{< M, w > | M \text{ is a TM that halts on } w\}$. $H_{TM} \notin R$ אפשר להוכיח ישירות או על ידי רדוקציה ל A_{TM}
- <u>שפה שאינה ב $R\varepsilon$ ואינה ב $Co - R\varepsilon$:</u> אפשר בקלות להגדיר שפה שמקבלת שתי מכונות ושתי מילים ומקבל אמ"מ מילה אחת שייכת לשפה של מכונה אחת, ומילה שנייה לא שייכת לשפה של מכונה שנייה. קל להראות שאם השפה הזו הייתה ב $R\varepsilon$ אז גם $\overline{A_{TM}}$ הייתה וגם אם המשלים שלה היה אז כנ"ל.
- <u>שיקולי עוצמות:</u> עבור $\Sigma = \{0,1\}$ נקבל ש Σ^* כלומר קבוצת כול המילים היא קבוצה בת מנייה, כי יש פונקציה חחע ועל על ידי הייצוג הבינארי, למספרים שלמים. מכאן נובע שקבוצת כול מכונות הטיורינג היא גם בת מנייה, כי יש העתקה חח"ע ממכונות טיורינג למחרוזות ולכן העוצמה קטנה מ $\aleph_0$ ולכן שווה לה כי יש אינסוף. נשים לב שאוסף כול הסדרות האינסופיות של ביטים היא לא בת מנייה (הוכחה של קנטור עם לכסון). נשים לב שיש העתקה חח"ע ועל משפות לסדרות בינאריות אינסופית על ידי פונקצית χ שהיא מתאימה לכול שפה סדרה שבמקום ה i יהיה בה 1 אם המילה ה i בסדר הלקסיקוגרפי שייכת לשפה ו אפס אחרת. לכן קבוצת כול השפות היא לא בת מנייה, ולכן משום שיש רק כמות בת מנייה של מכונות טיורינג, אין התאמת חחע ועל ולכן יש המון שפות שהן לא ב $\mathrm{R}\varepsilon$ וכתוצאה מכך גם במשלים.

<u>רדוקציות מיפוי:</u> הרעיון הוא בהנתן מכונה שפותרת בעיה אחת, נראה דרך להפוך אותה למכונה שפותרת את הבעיה השניה, והמעבר הזה חייב להיות משהו חשיב, כלומר בזמן סופי.

- <u>פונקציה חשיבה שלמה:</u> מכונת טיורינג שבהנתן קלט w כלשהו היא עוצרת ועל הסרט כתוב $f(w)$ בלבד.
- <u>פונקציה חשיבה חלקית:</u> מכונת טיורינג שעובר חלק המקלטים עוצרת ועל הסרט כתוב $f(w)$ בלבד, ועבור חלק מהקלטים לא עוצרת בכלל.
- <u>רדוקציה:</u> אם קיימת רדוקציה מ A ל B אז אפשר לפתור את A באמצעות היכולת לפתור את B. כלומר כול פתרון לבעיה B יוכל לשמש אותנו למצוא פתרון ל A. תכונה זו לא אומרת שדבר על הבעיות עצמן ועל הקושי בינהן, אלא רק על היחס בינן לבין עצמן. בשאיפה A לא תהיה קשה יותר מ B. ובפרט אם B כריעה אז גם A כריעה ואם A לא כריעה אז B לא כריעה.
- <u>Busy beaver:</u> לכול $n \in N$ נגדיר $S_n = \{all\ n\ state\ TM\ that\ halt\ on\ \varepsilon\}$ ונגדיר פונקציה חדשה $BB: N \to N$; $BB(n) = \max number\ of\ steps\ done\ by\ M \in S_n\ for\ input\ \varepsilon$. נשים לב שהכול מוגדר היטיב כי S_n היא קבוצה סופית כי יש רק כמות סופית של מכונות טיורינג בעלות n מצבים מעל א"ב מסוים, וכול $M \in S_n$ רצה כמות סופית של צעדים על ε כי

כולן עוצרות, לכן הפונקציה מוגדרת. נשים לב כי הפונקציה BB אינה חשיבה! כלומר לא קיימת מכונת טיורינג שלכול n תניב עבורנו את התוצאה בזמן סופי, כי אחרת היינו יכולים להכריע את בעיית העצירה על אפסילון, וזו בעיה לא כריעה. אולם נשים לב שהבעיה החסומה, כלומר פונקציה שתחזיר את ערכי הפונקציה עד n חסום כלשהו היא כריעה, כי אפשר לקודד את התשובה במצבים של המכונה, והערכים האלה, למרות שלא ידועים לנו קיימים, ולכן קיימת מכונה שמכריעה.

- <u>רדוקצית מיפוי:</u> פונקציה חשיבה $f:\Sigma^* \to \Sigma^*$ תקרא רדוקציה משפה A לשפה B אם $w \in A \leftrightarrow f(w) \in B$ ונסמן $A \leq_m B$. נשים לב כי $A \leq_m B \leftrightarrow \bar{A} \leq_m \bar{B}$, עבור אותה הפונקציה בדיוק. הפונקציה ממירה שאלה של שייכות לשפה A לשאלה של שייכות לשפה B. חשוב מאוד שפונקציית המעבר ניתנת לחישוב בזמן סופי. נשים לב כי הפונקציה ממש לא חייבת להיות חח"ע ועל!
- <u>הקשר בין כריעות לרדוקצית מיפוי:</u> $A \leq_m B \; and \; B \in R \; so \; A \in R$. נובע מיידית, על כול מילה ששואלים על שייכותה ל A נפעיל עליה את הפונקצית מיפוי ונריץ על המכריע של B וכך תהיה לנו תשובה לשייכות ל A. ובקונטרה פוזיציה אם $A \notin R$ אז גם $B \notin R$.
- <u>הקשר בין אינומרביליות לרדוקצית מיפוי</u>: $A \leq_m B$ וBi ניתנת למנייה אז גם A. כנ"ל גם לגבי הקונטרפוזיציה.
- <u>משפט Rice</u>: עבור תכונה לא טריוויאלית של שפות, כלומר עבור C תת קבוצה **ממש** , לא ריקה של שפות ב $R\varepsilon$, נסמן $L_c = \{< M > | L(M) \in C\}$ כלומר שפת כול המכונות טיורינג שהשפה שלהן מקיימת התכונה הזו. השפה $L_c \notin R$. ההוכחה היא על ידי בניית רדוקצית מיפוי לבעיית העצירה. התוכנה לא טריוויאלית ולכן יש שפה אחת לפחות L שמקיימת אותה, ובלי הגבלת הכלליות אפשר להניח שזו לא השפה הריקה , אחרת נסתכל על המשלים. קיימת מכונה שמקבלת את השפה הזו M_L. בהנתן קלט לבעית העצירה, כלומר מכונה ומילה $< M, w >$ נבנה מכונה חדשה D שלכול קלט x תריץ את M על w ולאחר מכן את M_L על y. לכן אם M עוצרת על w אז D תקבל בדיוק את L, אחרת השפה של D תהיה ריקה, והנחנו שהשפה הריקה לא מקיימת את התכונה הזו.

<u>Bounded Acceptance</u>: קיימות שתי השאלות הבאות: האם מכונת טיורינג מקבלת מילה תוך K צעדים? והשפה הזו כמובן כריעה, פשוט נריץ k צעדים ונבדוק. שאלה נוספת היא האם מכונת טיורינג מקבל מילה תוך שימוש בלכול היותר k תאים בזכרון? גם השאלה הזו כריעה! כי עבור זכרון חסום יש כמות סופית של קונפיגרציות שונות אפשריות, ולכן אם ניתן למכונה לרוץ יותר צעדים מכמות הקונפגרציות הזו אז מובטח שהיא חזרה על אחת פעמים ולכן לעולם לא תעצור.

<u>Re- completness</u>: בעיה תהיה re שלמה אם יש רדוקציה מכול בעיה ב re אליה, והיא עצמה ב re. למשל בעית הקבל היא כזו על ידי רדוקציה טיפשית.

<u>היסטורית חישוב והשימוש שלה להוכיח שלבדוק האם דקדוק חסר הקשר גוזר את Σ^* זו בעיה לא כריעה:</u> היסטוריית חישוב זה רצף סופי של קונפיגרציות שמופרדות ב # ומגדירות את הקונפיגרציות שעברה המכונה בזמן הריצה על מילה מסוימת. נאמר שהיסטוריה היא היסטוריה מקבלת אם הקונפיגרוציה הראשונה היא ההתחלתית, האחרונה מקבלת וכול המעברים היו חוקיים. נאמר שהיא דוחה אם כנ"ל אבל האחרונה דוחה. נשים לב שאם מכונה לא עוצרת על מילה מסוימת אז לא קיימת היסטוריה מקבלת ולא קיימת אחת דוחה גם.

- $ALL_{CFG} = \{G | G \; is \; a \; CFG \; and \; L(G) = \Sigma^*\}$. נראה שהשפה הזו לא כריעה על ידי שנראה $A_{TM} \leq_M \overline{ALL_{CFG}}$. הרעיון הוא שבהינתן מכונת טיורינג ומילה נבנה דקדוק חסר הקשר שייצר את כול המילים שהן לא היסטוריה מקבלת של המכונה, אם המכונה לא מקבלת את המילה אז הוא ייצר את כול המחרוזות בעולם, אם המכונה מקבלת את המילה אז יש לפחות היסטוריה מקבלת אחת שאותה הוא לא ייצר ולכן זה רדוקציה טובה. למעשה נבנה אוטומט מחסנית לא דטרמיניסטי שיקבל כקלט מחרוזת ויצטרך להכריע האם היא הסטוריה מקבלת

או לא, אם היא לא היסטוריה מקבלת אז יש קונפיגורציה אחת לפחות בקלט שמפרה את הכללים, והאוטומט ינחש באופן לא דטרמיניסטי איזה מהן לבחור. האוטומט יוודא שכול הקונפיגרציות אכן חוקיות, שהראשונה התחלתית, האחרונה מקבלת ושהמעברים חוקיים. בדיקת המעברים זה טיפה טריקי, ופתרון יפה הוא לרשום את היסטורית הריצה הפוך לסירוגין.

Linear bounded Automta: מכונת טיורינג שיכולה להשתמש במקום שהוא לינארי בגודל הקלט ולא יותר מזה, כלומר אין לה סרט אינסופי.

- A_{LBA} בעיית הקבלה עבור מודל זה, כלומר בהנתן מכונה ומילה האם היא מקבלת אותה זו בעיה כריעה! נבנה מכונה שתכריע את הביעה הזו על ידי סמלוץ ה LBA. אם הוא מנסה לחרוג מהגודל המוקצה לו אז המכונה תדחה, אחרת יש כמות סופית של קונפיגרציות אפשריות ולכן ניתן לסמלץ המכונה ליותר צעדים מהכמות הזו ומובטח שאם ה LBA לא עצר עד אז, אזי הוא תקוע בלולאה אינסופית ולכן אפשר לדחות.

Unristricted Grammer: בדומה לדקדוק חסר הקשר, רק שכאן יש חשיבות להקשר, כלומר ניתן להחליף מחרוזות שמכילות משתנים וגם אותיות למחרוזות חדשות. (לא ניתן להחליף מחרוזות שלא מכילות משתנים...) נוכיח שקבוצות כול השפות שקיים דקדוק כללי שגוזר אותן זה אותה הקבוצה כמו $R\varepsilon$. בהנתן דקדוק אפשר לייצר מכונה שתקבל את השפה שלו, כך שלמכונה יהיו שני סרטים, בסרט אחד היא תשמור את הקלט ובשני באופן לא דטרמינסטי תתחיל לגזור מילים. המכונה תקבל אם יש שוויון בין שני הסרטים. בהנתן מכונת טיורינג נגדיר דקדוק שכול הזמן יתחזק את מילה מהצורה WC כאשר w זה הקלט ו C זה הקונפיגורציה הנוכחית. במהלך הגזירה הוא יכול לשנות את הקונפיגרצה ממש כמו מכונת טיורינג ואז בסוף מקונפיגורציה מקבלת לעבור לאפסילון, וכך רק הקלט ישאר כמילה הגזורה.

סיבוכיות:

כעת נרצה לדעת כמה זמן לוקח למכונת טיורינג להכריע שפה מסוימת. את הזמן נמדוד כמספר הזזות הראש כתלות באורך הקלט n.

זמן זטרמניסטי: עבור מכונת טיורינג דטרמיניסטית בעלת סרט אחד M ופונקציה $t: N \to N$ נאמר ש M רצה בזמן $t(n)$ אם לכול קלט באורך n, M מבצעת לכול היותר $t(n)$ צעדים עליו. נסמן $DTime(t(n))$ להיות כול השפות שנתין להכריע בעזרת מכונה דטרמיניסטית בעלת סרט יחיד בזמן $O(t(n))$ (נשים לב שהמכונה צריכה לרוץ בזמן הזה גם עבור מחרוזות שהיא לא מקבלת).

מודל החישוב משנה! בשונה מחישיבויות שכול המודלים היו שקולים, כאן למודל יש חשיבות, ואכן ראינו שמכונת טיורינג עם כמה סרטים יכולה לחשב מהר יותר את השפה $\{0^n1^n\}$, כלומר עם שני סרטים אפשר בזמן לינארי, בעוד שעם סרט אחד אפשר ב $O(n \log n)$ בעזרת טריק יפה של השוואת הייצוג הבינארי, ובנוסף ראינו שאי אפשר בפחות, כי אחרת היה אפשר לנפח.

- ***MultiTape:*** נניח שקיימת מכונה *M* בעלת הרבה סרטים שמכריעה שפה *L* בזמן *t(n)* (שגדול שווה זמן לינארי) אז קיימת מכונה עם סרט אחד שמכריעה בזמן $t(n)^2$. זה נובע מהעובדה שכול צעד שהמכונה מרובת הסרטים עושה, המכונה עם סרט אחד תצטרך כנראה $t(n)$ צעדים לבצע אותו, כי היא צריכה לקרוא את כול התת סרטים שלה, שזה למעשה לעבור על כול הקלט מהתחלה עד הסוף, ואולי גם לדחוף את כול הקלט תא אחד ימינה, ולכן משום שאורך הסרט הכתוב הוא לכול היותר $t(n)$ אז לכול צעד שהמכונה המרובה עושה, המכונה עם סרט אחד תעשה לכול היותר $t(n)$ צעדים ולכן תכריע ב $t(n) * t(n) = t(n)^2$
- נשים לב שכול המודלים שקולים את כדי זמן פולינומיאלי, כלומר כול אחד יכול לסמלץ כול אחד בזמן פולינומיאלי, אבל לא בהכרח בזמן לינארי. ולכן נרצה לדעת אילו אלגוריתמים ניתן

להריץ בזמן פולינומיאלי כדי לא להיות תלויים במודל. בפרט מכונה לא דטרמיניסטית אינה מודל הגיוני, כי ההבדל בינה לביו שאר המודלים הוא אקספוננציאלי.

<u>זמן לא דטרמיניסטי:</u> נאמר שמכונת טיורינג לא דטרמינסטית רצה בזמן $f(n)$ אם לכול קלט מאורך n העומק המקסימלי של עץ החישוב, כלומר הענף הארוך ביותר הוא לכול היותר $f(n)$, בפרט כול ענף חייב להסתיים במצב מקבל / דוחה תוך פחות מ $f(n)$ צמתים.

- אם קיימת מכונה לא דטרמינסטית שמכריעה את השפה L ורצה בזמן $t(n)$ (במשמעות שהוגדר לעיל) אז קיימת מכונה דטרמינסטית שמכריעה את השפה ב $2^{O(t(n))}$. זה נובע מהיכולת לסמלץ מכונה לא דטרמינסטית על אחת שכן. המכונה הדטרמינסטית תעבור על כול הענפים האפשריים של העץ (כמו שראינו כבר) ותבדוק האם קיים ענף אחד מקבל. משום שהעומק של העץ חסום אז מספר העלים חסום על ידי $2^{O(t(n))}$ ולכן זה מספר הצעדים עד כדי הכפלה בקבוע פולינומיאלי של זמן הריצה עצמו.

<u>חשיבות הקידוד:</u> נשים לב שלאופן שבו אנחנו מקודדים את הקלט יש השפעה על איך שנפענח את זמן הריצה. אלגוריתם לינארי באורך הייצוג הבינארי הוא בעצם לוגריתמי במספר וכו'. לכן מקובל לעבוד עם אורך הייצוג הבינארי. אבל לא תמיד יש לזה חשיבות, למשל עבור גרפים.

<u>המחלקה P:</u> קבוצת כול השפות שקיימת מכונה דטרמיניסטית שמכריעה אותם בזמן פולינומיאלי. או במילים אחרות $p = \cup_{c\geq 0} DTIME(n^c)$, באופן כללי אלו יהיו בעיות שניתן לפתור בעולם האמיתי.

- <u>שפות לדוגמא:</u>
  - $Path = \{< G, s, t > | G\ is\ a\ direct\ graph\ with\ a\ path\ from\ s\ to\ t\}$. ניתן לפתור בזמן פולי עם אלגוריתם דומה לדיאקסטרה.
  - חישוב gcd של שני מספרים. אלגוריתם אוקלידס. פולי בייצוג הבינארי של המספר.

<u>המחלקה NP:</u> קבוצת כול השפות שקיימת מכונת טיורינג לא דטרמיניסטית שמכריעה אותם בזמן פולינומיאלי. (כלומר אורך המסלול העמוק ביותר...) $NP = \cup_{c\geq 0} NTIME(n^c)$, בפרט נקבל ש $P \subset NP$ כי מכונה דטרמיניסטית היא בפרט גם לא דטרמיניסטית.

- <u>מוודא:</u> עבור שפה L זו מכונת טיורינג V כך ש $L = \{w: V\ accepts\ (w, c) for\ some\ c\}$ כלומר קיימת מחרוזת ("עד") בעזרת העד הזה המוודא מסוגל לוודא שייכות לשפה L. את זמן הריצה של המוודא נמדוד כתלות במילה w. מוודא פולימויאלי זה מוודא שזמן הריצה שלו פולינומיאלי, ולכן נדרוש שאורך העד לא גדול מדיי. (לא לכול השפות יש מוודא פולינומיאלי, למשל המשלים של *HAMPATH*.
- <u>שקילות למוודא:</u> שפה היא NP אמ"מ קיים לה מוודא פולינומיאלי. אם היא ב NP אז יש מכונה לא דטרמיניסטית שמקבלת אותה, ולכן המוודא ישתמש כעד בבחירות הלא דטרמיניסטיות שנעשו, ולכן בהינתן הבחירות שנעשו הוא יוכל לעקוב אחר הענף שבו התקבלה המילה, ואורך הענף הוא פולינומיאלי. אם לשפה יש מוודא, אז מכונה לא דטרמיניסטית תנחש מחרוזת שתהיה העד, ואז תריץ את המוודא.

<u>המחלקה Co-NP:</u> מחלקת כול השפות שהמשלים שלהם ניתן לווידוא פולינומיאלי. נשים לב שלא יודעים האם היא שונה מהקבוצה *NP*, אבל יש שפות שנראה כאילו אין להם מוודא אבל למשלים שלהם יש, למשל המשלים של *Hampath*, קשה למצוא מוודא שיוכיח שאין מסלול, אבל אם יש מסלול ממש קל למצוא עד. $P \subset Co - NP$ כי p סגורה למשלים.

<u>המחלקה NP-Complete:</u> השפות "הקשות" ביותר במחלקה *NP*. הם השפות עם הסבירות הנמוכה ביותר שימצאו להן אלגוריתם פולינומיאלי, ואם ימצאו עבור אחת מהן זה יוכיח $P = NP$.

- <u>פונקציה חשיבה בזמן פולינומיאלי:</u> פונקציה $f:\Sigma^* \to \Sigma^*$ תקרא חשיבה בזמן פולינומיאלי אם קיימת מכונת טיורינג דטרמיניסטית שרצה בזמן פולינומיאלי שבהנתן קלט *w* בסוף הריצה על הסרט רשום $f(w)$
- <u>רדוקצה פולינומיאלית</u>: $A \leq_P B$ אם קיימת f חשיבה בזמן פולינומיאלי כך שמתקיים $w \in A \leftrightarrow f(w) \in B$. כלומר ניתן להמיר בזמן פולינומיאלי שאלה על שייכות ל *A* לשאלה על שייכות ל *B*. אם $A \leq_p B$ ומתקיים ש $B \in P$ אז $A \in P$. זה נובע מיידית, בהנתן מילה נמיר אותה לקלט של *B* בזמן פולי' נריץ את המכריע הפולי' של *B* ונחזיר תשובה.
- <u>הגדרה של המחלקה</u>: $L \in NP - Complete$ אם $L \in NP$ ולכול שפה $A \in NP$ קיימת רדוקציה פולי' כך ש $A \leq_P L$. ולכן מספיק שנמצא שפה אחת שהיא NPC שאפשר לפתור אותה בזמן פולי' והכול יקרוס.
- <u>להוכיח ששפה היא *NPC*</u>: צריך להראות תחילה שהיא ב *NP* ואז מספיק להראות רדוקציה ממנה לאיזשהי שפה *NPC* אחרת, וכך יהיה רדוקציה ממנה לכול שפה ב *NP*.
- <u>הוכחה ש $SAT \in NPC$:</u> לכול שפה $L \in NP$ קיימת מכונה לא דטרמינסטית *M* שמכירעה אותה בזמן n^c, ולכן לכול קלט *w* קיימת טבלה בגודל $n^c \; x \; n^c$ שמתארת את הריצה, כאשר השורה ה *i* היא הקונפיגורציה של המכונה בשלב ה *i* שלה הריצה, כאשר השורה הראשונה היא הקונפיגורציה ההתחלתית. נבנה נוסחא שתתנהג בדיוק כמו הטבלה כך שהיא ספיקה אמ"מ הטבלה מתארת ריצה מקבלת. המשתנים שלנו יהיו מהצורה $x_{i,j,s}$ כאשר כול משתנה יהיה 1 אמ"מ בשורה ה *i* בעמודה ה *j* נמצא התו *s*. כלומר יהיו לנו *s* משתנים לכול תא בטבלה (זה עדיין פולי'). הנוסחא שלנו כעת תצטרך לוודא שאכן הקונפיגורציה הראשונה היא התחלתית, כול אחת בדרך היא חוקית, האחרונה מקבלת, והמעברים חוקיים, ולעשות *and* בין כול הדברים הללו. אזי הנוסחא והטבלה מגדירים אחד את השני באופן חח"ע. נשים לב שלבדוק שהראשונה אכן התחלתית זה קל. לבדוק שכולן חוקיות זה לבדוק שלכול תא יש לפחות אחד מהמשתנים שלו שהם 1 וכול השאר אפס. לבדוק שבדרך קילבנו זה גם פשוט. לבדוק שהמעברים חוקיים זה אפשרי משום שמעברים במכונת טיורינג הם תמיד לוקלים.
- <u>שפות לדוגמא</u>:
  - $\boldsymbol{A_{NP}} = \{< M, x, 1^n > | M \; is \; a \; TM \; and \; \exists c \in \Sigma^* s.t \; M(x,c) acc \; in \; n \; steps\}$. זו השפה הטריוויאלית שמוכיחה שהמחלקה לא ריקה. (הרדוקציה לכול שפה אחרת תהיה להעביר את המוודא...)
  - $\boldsymbol{SAT} = \{$נוסחא ספיקה $\phi\}$. נוסחא בוליאנית היא ביטוי שמכיל משתנים בוליאנים ופעולות בוליאניות כמו *and,or*. נוסחא ספיקה אם קיימת השמה למשתנים שמניבה ערך אמת לנוסחא.
  - $\boldsymbol{3SAT}$: בדומה ל *SAT* רק שלנוסחא יש צורה מיוחדת. נוסחא היא מצורת *3CNF* אם הנוסחא מורכבת מסוגריים (*clauses*) המכילים שלושה ליטרלים (משתנה או השלילה שלו) כאשר בתוך הסוגרים יש *or* ובין הסוגרים יש *and*. (נשים לב ש *1SAT* וגם *2SAT* הם פולי')
  - $\boldsymbol{Clique} = \{< G, k > | G \; contains \; a \; clique \; of \; size \; k\}$ ההוכחה על ידי רדוקציה ל *3SAT* על ידי בניית גרף שהצמתים בו מחולקים לשלשות לפי השלשות של הליטרלים בנוסחא. נחבר כול שני צמתים בקשת מלבד צמתים באותו ה *clause* ולא נחבר ליטרלים והשלילה שלהם.
  - $\boldsymbol{independentSet} = \{< G, k > | G \; contains \; an \; IS \; of \; size \; k\}$ נשים לב שיש רדוקציה פשוטה מאוד ל*clique*. בהנתן גרף שיש בו קליקה אז בגרף המשלים יש *IS*. לחשב גרף משלים זה פולינומיאלי.
  - $\boldsymbol{Hampath} = \{< G, s, t > | G \; contains \; a \; Hamilton \; path \; between \; s \; and \; t\}$ (וגם זה גרף מכוון). נראה ש $3SAT \leq_P HamPath$ על ידי שבהנתן נוסחא נבנה גרף מכוון שיכיל מסלול המילטוני אמ"מ הנוסחא ספיקה. בגרף כול *clause* יקבל

צומת, וכול משתנה יקבל תת גרף בצורת יהלום, כאשר מרכז היהלום מורכב מזוגות של צמתים שכול זוג מייצג *clause* אחר. אם השמתנה מופיע ב*clause* מסוים נחבר את הקודקוד של ה *clause* הזה עם המקום הנכון ביהלום, אם הוא מופיע בשלילה נחבר בכיוון ההפוך. ככה אם יש מסלול אז בכול יהלום צריך לבחור כיוון ללכת בו, כיוון אחד יגדיר שהמשתנה הזה מקבל ערך אמת, וכיוון שני אומר שהוא מקבל ערך שקר, וזה מבטיח שבכול המופעים שלו הוא יהיה באותו הערך כי אי אפשר לשנות את הכיוון של המסלול באמצע, אחרת הוא לא יהיה המילטוני. בנוסף כדי שיהיה מסלול המילטוני מובטח שעברנו גם בצמתים של ה *clause* וזה אומר שהיה לפחות משתנה אחד בכיוון המתאים, ולכן בכול *clause* היה לפחות ליטרל אחד שערכו אמת.

- $\mathbf{HamCycle} = \{< G > | G\ as\ a\ HamCycle\}$ כאשר *HamCycle* זה למעשה מסלול המילטוני שמתחיל ומסתיים באותו קודקוד. רדוקציה פשוטה למסלול המילטוני על ידי חיבור הצומת ההתחלתית עם הצומת הסופית בעזרת קודקוד נוסף חדש.
- $\boldsymbol{TravelingSalesman}$ בהנתן גרף מלא עם משקלים חיוביים על הקשתות, מצא מעגל המילטוני מאורך לכול היותר *k*. אפשר בקלות להראות רדוקציה למציאת מעגל המילטוני, על ידי בהנתן גרף נייצר גרף מלא שהקשתות שהיו קיימות במקורי יקבלו משקל 1 ואלא שלא היו יקבלו משקל 2, ונבחר *k* להיות מספר הקשתות שהיו במקור.
- $\boldsymbol{Partition} = \{< x_1, \dots x_n > \exists S \subset [1, \dots, n]: \Sigma_{i \in S} x_i = \Sigma_{i \notin S} x_i\}$ רדוקציה לסאבסטסאם על ידי הוספת שני איברים חדשים, האחד זה סכום כול האיברים ועוד הסכום החלקי, והשני זה פעמיים סכום כול האיברים פחות הסכום החלקי.
- $\boldsymbol{SubsetSum} = \{< x_1, \dots, x_n, t > | \exists \{y_1, \dots, y_k\} \subset \{x_1, \dots, x_n\}: \Sigma y_i = t\}$ נובע מבניית טבלה שלכול משתנה יהיו שתי עמודות ולשים אחדות במקומות הנכונים... ורדקוציה ל *3SAT*.
- $\boldsymbol{BinPacking} = \{< S, B, m >: \exists partition\ s_1, \dots, s_m\ s.t\ \forall j\ \Sigma_{s \in S_j} s \leq B\}$ ועושים רדוקציה ל *partition* כאשר כמות הקבוצות היא שתיים והחסם הוא מחצית סכום כולם.
- $\boldsymbol{IntegerPrograming}$**:** בהנתן סט של אי שוויונים עם מקדמים שלמים, האם קיים פתרון בשלמים למערכת. נשים לב שמכלל קרמר לפתרון מערכות לינאריות אם יש פתרון האורך שלו פולינומיאלי, ולכן קיים עד למוודא פולי ולכן זה ב *NP*. בקלות אפשר לעשות רדוקציה ל *3SAT* על ידי הגדרת מערכת משוואת שתהיה כמו בוליאנית.
- $\boldsymbol{VertexCover}$ גרף וגודל *k* כך שקיים לגרף כיסוי על ידי *k* קודקודיים, כלומר כול הקשתות מתחילות או מסתיימות בלפחות אחד מהקודקודיים בקבוצה מגודל *k*. (נשים לב שעבור *k* שווה מספר הקודקודיים יש פתרון טריוויאלי, הגרף כולו). מראים רדוקציה ל *IS*.

<u>היררכית זמני הריצה:</u> לכול פונקציה $t(n)$ שפויה, כלומר שלוקח פחות מ $t(n)$ זמן לחשב אותה, וגם $t(n) \geq nlog\ n$ אז קיימת שפה שניתן לחשב ב $O(t(n))$ אבל לא ב $o\left(\frac{t(n)}{logn}\right)$. כלומר עוד זמן מוסיף ממש עוד שפות. ההוכחה היא על ידי הגדרת שפה מסוימת שמקבלת כקלט מכונת טיורינג ואחרי זה רצף אפסים. היא בודקת האם הקידוד של המכונה לא ארוך מ $\log(t(n))$ ואז מריצה את המכונה על הקלט לכול היותר $t(n)/\log(t(n))$ צעדים ודוחה אם המכונה לא סיימה עד אז, ואם כן סיימה תחזיר ההפך. נשים לב שהמכונה הזו מכריעה את השפה שלה בזמן $t(n)$ כי נוכל להעתיק איתנו במהלך הריצה על סרט יחיד את שני הסרטים הקצרים יותר (המונה והקידוד של המכונה שהם לוגריתמים).

נניח בשלילה שניתן להכריע את השפה בפחות בזמן אז נגיע לסתירה מאיך שהגדרנו את השפה, נקבל שהמכונה המהירה יותר לא מקבלת את אותה השפה.

NP-Hard: שפה תקרא *NP* קשה אם לכול שפה $L \in NP$ קיימת רדוקציה פולינומיאלית אליה.

Co-NP Hard: שפה תקרא $Co-NP\ Hard$ אם לכול שפה ב $Co-NP$ יש רדוקציה פולי אליה.

בעיות: נשים לב שעד עכשיו התעסקנו בבעיות הכרעה, עכשיו נרצה ממש להיות מסוגלים לתת פתרון לבעיות. לכן נגדיר בעיה להיות $T \subset \Sigma^* \chi \Sigma^*$ כלומר אוסף של זוגות מילים, כאשר פותר לבעיה *S* יהיה אלגוריתם כך שלכול $x \in \Sigma^*$ יתקיים $S(x) = y \leftrightarrow (x,y) \in T$, ועבור איברים שאין להם זוג אז התשובה שלו לא מוגדרת. נשים לב שבעיות הכרעה שקולות לזוגות של מילים בשפה ואחד למילים ששיכות לה ואפס אחרת.

רדוקצית Karp: נאמר ששפה *L* ניתן לרדוקצית *karp* לבעיה *T* אם קיים אלגוריתם *D* פולינומיאלי שנעזר באורקל, כך שלכול *S* פותר של הבעיה *T*, מתקיים ש *D* בעזרת פניות אורקל ל *S* מכריע עבור השפה *L*. נאמר שבעיה היא *NP* קשה אם יש רדוקצית *karp* ממנה לכול שפה ב *NP*.

בעיות חיפוש: בהנתן שפה מסוימת, ומילה, בעיית החיפוש היא למצוא את העד שמוכיח את שייכות המילה לשפה. עבור שפות *NPC* הבעיה הזו קשה ממש כמו להכריע את השפה עצמה (יש רדוקצית *Karp* לשני הכיוונים). נשים לב שבהנתן מכריע לשפה כמו *SAT,Clique* זה ממש פשוט למצוא את העד, על ידי הקטנת הבעיה באחד, ושאילת המכריע האם עדיין יש פתרון. זה תכונה של שפות *NPC*.

בעיות אופטימזציה: קשה יותר אפילו מבעיות החיפוש. בהנתן בעיה, לא רק תמצא את הפתרון, אלא תמצא את הפתרון הקטן / גדול ביותר.

אז איך בכול זאת מתמודדים עם NP? אפשר למצוא כול מיני קירובים, שיפורים, ופתרונות ששואפים להיות הפתרון הטוב ביותר. למשל מציאת ה *VC* המינמלי, אז ניתן למצוא עד פי שתיים מהמינמלי על ידי באופן נאיבי כול פעם לקחת קשת אחת יחד עם הקודקודים שלה ולמחוק אותה מהגרף יחד עם הקודקודים שלה ולהמשיך. אפשר גם לפתור בעזרת אקראיות מסוימת ולקוות לפתרון טוב בתוחלת. בנוסף אפשר לפתור בעיות עבור פרמטרים קבועים, כלומר למצוא קליקה בגודל 5, או 6.. קושי הבעיה תלויה בגודלה.