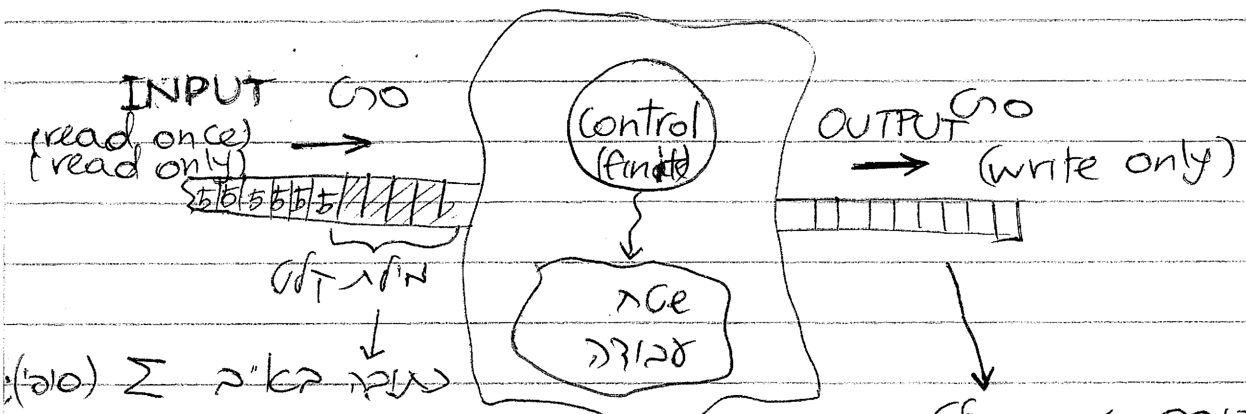
הגדרות בסיסיות

* מודלים חישוביים שנעסוק בהם בקורס זה, ניתן להציג סכמטית כ:



כתובה בא"ב Σ (סופי): $\not b$ (רווח) סימן מיוחד, הממלא את כל הסרט פרט למילת הקלט; $\not b$ הראשון מסמן סיום הקלט.

בקורס זה הפלט, בדרך כלל, יהיה 0/1, משמע "הקבל" או "נדחה".

* שפות מעל Σ:

אות ---> מילה ---> שפה ---> מחלקה של שפות

$\in \Sigma$ — Σ^* סופית-ε — $\subseteq \Sigma^*$ (אוסף סופי או אינסופי של מילים)

מילה מיוחדת: ε, מילה ריקה

שפות מיוחדות: $\emptyset$: שפה ריקה; Σ^*: הכל.

שים לב: $\emptyset \neq \varepsilon$!

* מודל M מקבל (accepts) מילה w אם הריצה של M על w מסתיימת ב"הקבל". M דוחה (rejects) w אם הריצה מסתיימת ב"נדחה". שים לב: אם הריצה לא מסתיימת, w לא מקבל ולא נדחה!

* מודל M מקבל שפה L אם

$$L = \{ w \in \Sigma^* \mid M \text{ מקבל את } w \}$$

L נקראת מוגדרת ע"י M באופן חד-ערכי; לפעמים נסמנה ב L_M.

* מודל M מכריע או מזהה (decides) שפה L אם

1. M מקבל את L;
2. M דוחה כל $w \notin L$.

בחלק מהמודלים בהם נדון שני המושגים (מקבל ומכריע) מזדהים; בחלקם יהיה ביניהם שוני משמעותי.

מטרת הקורס: הכרה של מודלים בסיסיים וחשובים:

אוטומט סופי → אוטומט מחסנית → מכונת טיורינג

אוטומט סופי: מודל מצומצם מאוד

אוטומט מחסנית: מודל בכוח מצומצם ובעל שימושים רבים במדעי המחשב

מכונת טיורינג: המודל האולטימטיבי

כמו כן, הכרה של התכונות של מחלקות של שפות המתקבלות ע"י המודלים הנ"ל. בפרט, נלמד "לתכנת" במודלים הנ"ל, וגם להוכיח שפות מסוימות לא ניתנות לזיהוי.

פרק 1

אוטומטים סופיים

אוטומט סופי (דטרמיניסטי, בקיצור), DFA, יש להתייחס כאל מחשב בעל זכרון סופי בלבד. בשביל לתכנת במחשב כזה, אנו חייבים בכל שלב ושלב לתמצת את החלק של INPUT שקראנו במספר סופי (קבוע מראש) של ביטים.

(*) הגדרה פורמלית:

Q: המצבים (מס' סופי קבוע), כאשר:

צריך להתייחס אליהם כאל כל הקונפיגורציות האפשריות של הזכרון

$q_0 \in Q$ הוא מצב מיוחד הנקרא מצב התחלתי, ו

$F \subseteq Q$ היא תת-קבוצה של מצבים הנקראים מצבים מקבלים.

כמו כן, $\delta : Q \times \Sigma \to Q$ היא פונקצית מעבר (מוגדרת היטב).

החישוב על מחרוזת $\sigma = \sigma_1 \sigma_2 \ldots \sigma_k$ מתבצע באופן הבא:

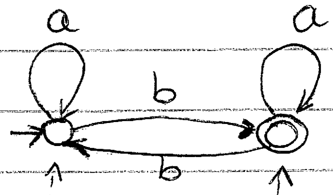
$$q_0 \xrightarrow{\sigma_1} \underset{\delta(q_0,\sigma_1)}{\underset{\parallel}{q^{(1)}}} \xrightarrow{\sigma_2} \underset{\delta(q^{(1)},\sigma_2)}{\underset{\parallel}{q^{(2)}}} \ \ldots \ \xrightarrow{\sigma_k} \underset{\delta(q^{(k-1)},\sigma_k)}{\underset{\parallel}{q^{(k)}}} .$$

אם $q^{(k)} \in F$ נאמר כי σ מתקבלת; בכל מקרה אחר, נאמר כי σ נדחית. אין, אם כן, במודל הזה הבדל בין לקבל ולעצור.

שים לב כי אם $q_0 \in F$, אזי ε בהכרח מתקבלת.

הצגה אלטרנטיבית, המתאים במיוחד לאוטומטים קטנים, היא הצגה גרפית:

דוגמא:
$\Sigma=\{a,b\}$



סימון למצב התחלתי

סימון למצב מקבל

החיצים מתארים את פונקציית המעבר δ.

מה מקבל האוטומט הנ"ל? קל להשתכנע כי אלו בדיוק כל המחרוזות w מעל $\{a,b\}$, שמספר ה-b-ים בהן הוא אי-זוגי.

(*) נלך בדרך הפוכה: נתחיל מהשפה, וננסה שנמצא לה אוטומט סופי.

(+) דוגמא 1

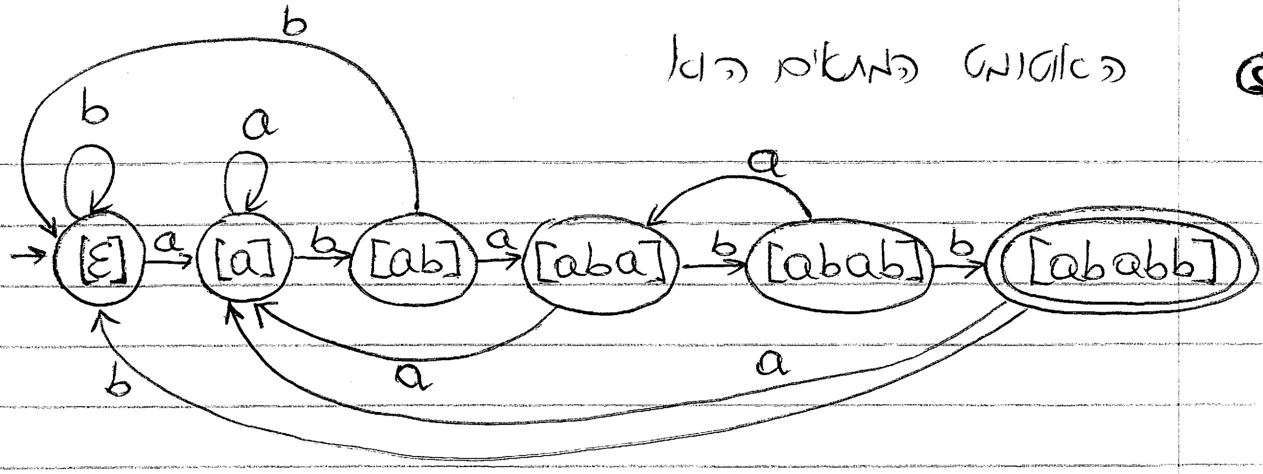
$L=\{w\in\Sigma^* \mid w \text{ מסתיימת ב- } ababb\}$, $\Sigma=\{a,b\}$

פתרון (הידוע מקורס "אלגוריתמים"):

+ למצבים שמות של רישות של ababb;

$\delta(q,\sigma)=$ הסיפא הארוכה ביותר של "$q\sigma$" (שרשור) שהיא גם רישא של ababb +

5 האוטומט המתאים הוא



+ דוגמא 2:

$\Sigma=\{0,1,2,\dots,9\}$

L={ המחרוזות המייצגות ייצוג עשרוני של מספרים המתחלקים ב7 (m.s.b. first) }

φ, 0014 מתקבל, 613 נדחה.

לצורך הפתרון, נזכר באלגוריתם של חלוקה ב7 הנלמד בבית ספר:

$$\begin{array}{l|l} 17482104\dots & 7 \\ \hline 14 & 249\dots \\ \underline{}34 & \\ 28 & \\ \underline{}68 & \\ 63 & \\ 52 & \end{array}$$

בכל שלב ושלב המידע היחיד שאנו שומרים על הכל שראינו עד עכשיו, הוא מס' ב [0..6] - השארית בחילוק ב 7. זה מצב אידאלי לשימוש באוטומט סופי:

$Q=\{0,1,\dots 6\}$; $\delta(q,\sigma)=q\cdot 10+\sigma \bmod 7.$

"0" - התחלתי, וגם מקבל.

+ דוגמא 3

$\Sigma=\left\{\binom{0}{0},\binom{0}{1},\binom{1}{0},\binom{1}{1}\right\}$. עבור $w\in\Sigma^*$ נגדיר w^{up} w_{low} באופן הבא:

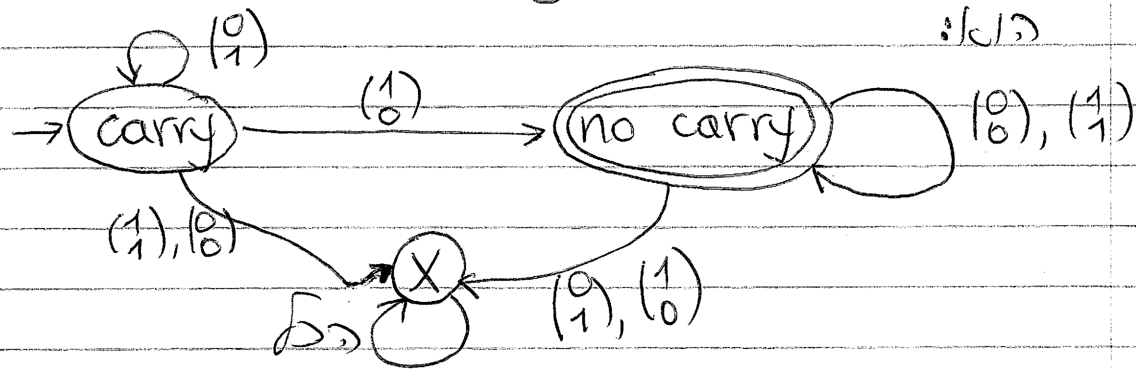
$w=\begin{pmatrix}0&1&0&1&1\\1&1&0&0&0\end{pmatrix}$. $01011=w^{up}$, $11000=w_{low}$

⑥

נגדיר $L=\{w\in L \mid w^{up} = w_{low}+1\}$,
כאשר אני מתייחסים ל-w^{up}, w_{low} כקידודים בינאריים של מספרים (הפעם l.s.b. first).

לדוגמא, $\begin{pmatrix}0&1\\0&0\end{pmatrix}\in L$, $L\ni\begin{pmatrix}1&0&0&0\\0&1&1&1\end{pmatrix}$.

ננסה לחשוב את הבעיה באופן הבא: יש שני מצבים עיקריים, "יש carry" או "אין carry", והן מצב "לא מסתדר". בהתחלה יש סמוי carry (הרי אנו מבצעים +1); ה carry מועבר עד אשר אנו נתקלים ב $\binom{1}{0}$; מרגע זה רק $\binom{0}{0}$ ו $\binom{1}{1}$ באים בחשבון.
על כן, כל מה שעלינו לזכור זה האם עדיין לא פספסנו, והאם אנו סוחבים או carry או לאו. האוטומט המתאים הוא:



⊕ דוגמא 4
$\Sigma=\{a,b\}$, $L=\{w\in\Sigma^* \mid$ "a" האות הרביעית מסוף היא$\}$.

ברור כי מספיק לזכור את ארבעת האותיות האחרונות שראינו. לכן,
$Q=\{(b_1b_2b_3b_4) \mid b_i\in\Sigma\}$
פונקצית המעבר המתאימה:
$(b_1b_2b_3b_4)\xrightarrow{\sigma}(b_2b_3b_4\sigma)$.
מצבים מקבלים: כאלה ש $a=b_1$.
מצב התחלתי: קל להשתכנע כי (bbbb) מתאים.

דוגמא 5

כמו בדוגמא 2 (מספרים עשרוניים המתחלקים ב-7), אך הפעם l.s.b. first.

$\boxed{L_5 = L_2^r}$

L_5: שפה של 5
L_2: השפה של 2
r: אופרטור reverse

פתרון א: לו יכולנו לעשות הרצה "בנסיגה", צ"ל להתחיל במצב המקבל של 2, לעבור בכל פעם מעין באות, ולסיים במצב ההתחלתי של 2, ה"ינו מעבירים בדיוק את L_5. ואכן, התכונה עובדת! הסיבה העיקרית להצלחה היא שפונקצית המעבר של 2 היא הפיכה, לכן הנסיגה תמיד מוגדרת באופן חח"ע:

אם $\delta_2(q,\sigma) = q\cdot 10 + \sigma \mod 7 = p$,
צריכה לקיים ש-
$$\delta_5(p,\sigma) = q \Rightarrow \delta_5(p,\sigma) = (p-\sigma)\cdot 5 \mod 7$$
(הסיבה להופעת 5 היא העובדה $10\cdot 5 = 1 \mod 7$)

כמו דונם, המצב ההתחלתי והמס"מ הוא "0".

פתרון ב: ננסה ללכת בהגיון הרגיל שלנו, ולהשתדל לתמצת את כל מה שקראנו עד עתה בזכרון קבוע.

$\leftarrow$
...471321

ברור כי עלינו לזכור את השארית mod7, הפעם זה לא מספיק:

$$1 \rightarrow 21 \rightarrow 321 \rightarrow 1321 \rightarrow 71321 \rightarrow \ldots$$

$$1 \rightarrow 2\cdot 10+1=0 \rightarrow 3\cdot 10^2+0=6 \rightarrow 1\cdot 10^3+6=5 \rightarrow 7\cdot 10^4+5=5 \rightarrow \ldots$$

צריך לכעורה לספור את מספר הצעדים שעשינו...

פתרון:

$$Q = \{ (X,Y) \mid 0 \le X \le 6,\ 1 \le Y \le 6 \}$$

שארית mod7 של הרישא הנוכחית

שארית של 10 הנוכחי mod 7

$(X,Y) \xrightarrow{\sigma} (Y\cdot\sigma + X \bmod 7,\ Y\cdot 10 \bmod 7)$;

מצבים מקבלים: $X=0$;

מצב התחלתי: $(0,1)$.

נשים לב שבנינו אוטומט שונה ($42 = 7\cdot 6$ מצבים כנגד 7 מצבים, לדוגמא) המקבל אותה השפה!

---

* פעולות על שפות ופעולות על האוטומטים

(* to be cont. הערה חשובה *)

נניח כי L_1, L_2 הן שפות רגולריות, ו- A_1, A_2 הם האוטומטים המקבלים אותם. אזי

1. $\overline{L_1}$ היא רגולרית.
2. $L_1 \cup L_2$ היא רגולרית.
3. $L_1 \cap L_2$ היא רגולרית.

(1.) ד' מיידי: נקח A_1, ונהפוך מצבים מקבלים ללא מקבלים, ולהפך. A_1' המתקבל מקבל את $\overline{L_1}$.

(2.), (3.) היינו רוצים להריץ את הקלט על A_1, A_2 במקביל, ולהחליט האם לקבל או לא בסוף הריצה.

אכן ניתן לעשות את זה!

נניח לנוחות כי, בה"כ ל-A_1 ול-A_2 יש אותו א"ב Σ. (אם אין הדבר כך, נוסיף ל-A_1 (ובהתאם, ל-A_2) מצב חדש - בור - ונגדיר

$\delta_1(q,\sigma) =$ בור ; $\sigma \notin \Sigma_1 , \sigma \in \Sigma_2$; $\delta_1($בור$,\sigma) =$ בור לכל σ).

נגדיר אוטומט מכפלה $A_1 \times A_2$ באופן הבא:

- מצבים: $Q_1 \times Q_2$
- מעברים: $(q^1, q^2) \xrightarrow{\sigma} (\delta_1(q^1,\sigma), \delta_2(q^2,\sigma))$
- מצב התחלתי: (q_0^1, q_0^2).

מצבים מקבלים נגדיר באופן המתאים לשאלה:

$F = \{ (q^1,q^2) \mid q^1 \in F_1 \wedge q^2 \in F_2 \}$ נקבל $L_1 \cap L_2$

$F = \{ (q^1,q^2) \mid q^1 \in F_1 \vee q^2 \in F_2 \}$ נקבל $L_1 \cup L_2$.

<u>שימו לב</u>: הבנייה כללית לחלוטין, מוגדרת לכל זוג אוטומטים/ שפות!

אפיון שפות רגולריות ואוטומט מינימלי

(* to be continued! *)

מטרתנו להציג קריטריון של רגולריות/אי רגולריות של שפה נתונה, אשר אינו בונה במפורש את האוטומט המתאים. תוך כדי זה, נקבל גם הערכה הדוקה של מס' המצבים הדרוש כדי לקבל את השפה.

נתחיל מהבחנה הבאה. יהי A אוטומט סופי כלשהו, q מצב כלשהו ב A. נניח כי הריצה ב A של שתי מחרוזות $x,y \in \Sigma^*$ מסתיימת ב q. אזי, לכל $w \in \Sigma^*$, גם xw, yw יסתיימו באותו מצב, ובפרט לא יתכן כי $xw \in L_A$, $yw \notin L_A$.

זה מצדיק הגדרה הבאה:

* תהי L שפה מעל Σ. יחס שקילות $\equiv_L$ מוגדר על Σ^* באופן הבא:

$x \equiv_L y \iff \forall w \in \Sigma^*$ xw, yw שתיהן ב L, או שתיהן לא ב L.

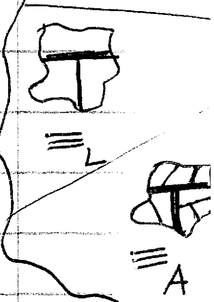
* **משפט**: L רגולרית אם ורק אם מספר מחלקות השקילות של $\equiv_L$ הוא סופי.

$\boxed{\Leftarrow}$ נניח כי L רגולרית, ויהי A אוטומט המקבל אותה. נגדיר יחס $\equiv_A$ על Σ^* באופן הבא:

$x \equiv_A y \iff$ הריצות של x ושל y מסתיימות באותו המצב

לפי ההבחנה, $x \equiv_A y \Rightarrow x \equiv_L y$, ועל כן היחס $\equiv_A$ הוא עידון של היחס $\equiv_L$, ובפרט מס' מחלקות השקילות ב-$\equiv_A$ $\geq$ מזה של $\equiv_L$. כיוון שמס' מחלקות השקילות תחת $\equiv_A$ הינו מס' המצבים ב-A, גמרנו.

העדנה:



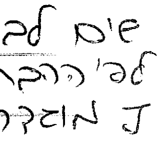
$\boxed{\Leftarrow}$ נניח כי ל-$\equiv_L$ מספר סופי של מחלקות שקילות; נקרא להם $K_1, K_2, \ldots, K_r$. לפי ההגדרה, ה-K_i-ים זרים בזוגות, ואיחודם הוא Σ^*.

⊕ הבחנה: אם $K_i \ni x, y$, ו-$K_j \ni x\sigma$ עבור $\sigma \in \Sigma$ מסוימת, אזי גם $K_j \ni y\sigma$ – אותו j!

אכן, אם $x\sigma$ ו-$y\sigma$ היו שייכות למחלקות שונות, היה ק"ם $w \in \Sigma^*$ φ ש אחד מבין $x\sigma w$, $y\sigma w$ היה שייך ל-L, והשני לא – בסתירה להגדרת $\equiv_L$ ובחירת y, x.

⊕ באופן דומה, אם $K_i \ni x$ הוא ב-L, אזי כל $K_i \ni y$ הוא גם ב-L, ולהפך. כעת, כיוון ש-$x\varepsilon$ ו-$y\varepsilon$ שניהם שייכים או שניהם לא שייכים לפי הגדרת $\equiv_L$.

כ"א, $\equiv_L$ הוא עידון של



נבנה אוטומט A_L באופן הבא:

- מצבים: $q_1, \ldots, q_k$ (המתאימים ל-$K_1, \ldots, K_r$)
- פונקצית מעבר:

$$\delta(q_i, \sigma) = q_j, \quad x \in K_i \Rightarrow x\sigma \in K_j$$

שים לב: לפי ההבחנה ⊕, δ מוגדרת היטב!

- מצב התחלתי q_i, ש-$\varepsilon \in K_i$.
- מצבים מקבלים: q_j מקבל אמ"מ $L \supseteq K_j$.

מאופן בנית האוטומט ברור כי $w \in K_i \Leftrightarrow$ החישוב על w מסתיים ב-q_i;

לכן, כיוון ש-K_i כולו $\subseteq L / \bar{L}$ האוטומט נותן תשובה נכונה ▨.

<u>הערה:</u> בפועל הוכחנו יותר ממה שטענו. הראינו כי כל אוטומט A המקבל את L חייב להיות בעל $\geq |\equiv_L|$ מצבים. יתר על כן, כל אוטומט A המקבל את L שיש בו כזה מס' של מצבים. A כזה הוא יחיד: כי $\equiv_L$ ו $\equiv_A$ חייבים במקרה זה להתלכד, ובמקרה זה פונקציית המעבר מוגדרת לחלוטין ע"י $\equiv_L$. A כזה נקרא <u>האוטומט המינימלי</u> של L.

$|\equiv_L|$: מס' מחלקות השקילות ב $\equiv_L$

<u>דוגמא א':</u> $L = \{0^n1^n \mid n\in \mathbb{N}\}$.
ראינו שעבור השפה הזאת, יש ב $\equiv_L$ אינסוף מחלקות, ולכן L <u>איננה רגולרית</u>. אכן כל 0^i שייך למחלקה יחודית:
$$0^i \not\equiv_L 0^j \ (i\neq j) \quad \text{since} \quad 0^i1^i \in L,\ 0^j1^i \notin L.$$

<u>דוגמא ב':</u> $L = \{w\in \{0,1\}^* \mid \text{הסימן הרביעי מהסוף הוא } 1\}$.
כפי שראינו, יש אוטומט A בעל 16 מצבים המקבל את L. נראה שהאוטומט הזה הוא מינימלי, כלומר יש ב $\equiv_L$ בדיוק 16 מחלקות. מספיק להראות 16 מילים כך שכל שתיים מהם אינם שקולים $\equiv_L$. המילים הנ"ל הם $\{b_1b_2b_3b_4\}_{b_i\in\{0,1\}}$. כל שתיים מהם שונים, כי, לדוגמא
$$0101 \not\equiv_L 0001 \quad \text{since} \quad 01010 \in L,\ 00010 \notin L.$$

לפני שנמשיך הלאה, ניתן קריטריון נוסף לרגולריות.

<u>למת השאיבה (ניפוח)</u>

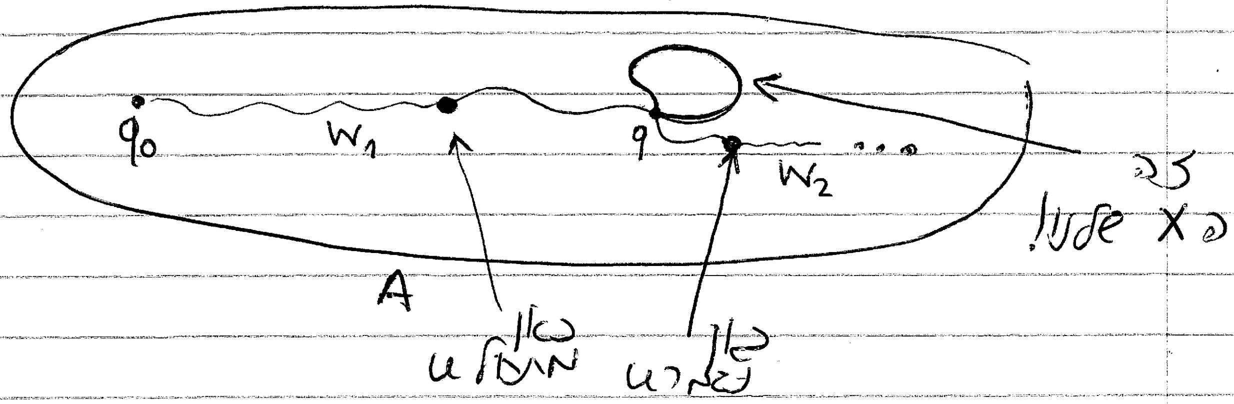
בהינתן שפה L, ומילה $w\in\Sigma^*$ מהצורה $w_1\cdot x\cdot w_2$, x נקרא <u>קטע ניפוח</u> אם <u>כולם</u> $\{w_1\cdot x^i\cdot w_2\}_{i=0}^{\infty}$ שייכים ל L או אף אחד לא שייך ל L.

$x^0=\varepsilon$: גם שאיבה

<u>משפט:</u> תהי L שפה רגולרית, המתקבלת ע"י אוטומט A בעל k מצבים. אזי, <u>כל</u> קטע רציף u, של <u>כל</u> $w\in\Sigma^*$ מכיל <u>קטע ניפוח</u>.
$|u|\geq k$ [illegible]

$$w = w_1 \underbrace{\overbrace{u}^{x \neq \varepsilon}}_{|u| \geq k} w_2$$

<u>הוכחה:</u> נתבונן בהרצה של w ב-A:



כיוון ש-$|u| \geq k$, וב-A יש $\leq k$ מצבים, בזמן הרצת u מצב מסוים חייב היה להופיע פעמיים. נקרא לו q; x יהיה מוגדר כתת-מחרוזת המתחילה ומסתיימת ב-q. ברור כי שאיבה/ניפוח של x לא ישנו את המשך הריצה, ובפרט לא ישפיעו על קבלה/דחיה. ∎

<u>דוגמא</u> לשימוש בלמת השאיבה/ניפוח:

$L = \{ w^R w \mid w \in \{0,1\}^* \}$ השפה אינה רגולרית.

הוכחה: נניח בשלילה ש-L רגולרית, וניתנת לקבלה ע"י אוטומט בעל k מצבים. נתבונן במילה $0^k 1^2 0^k \in L$. לפי למת השאיבה/ניפוח, הקטע הראשון של אפסים מכיל קטע ניפוח. אבל מיד ברור כי השאיבה של הקטע הנ"ל פוגעת בהשתייכות ל-L. סתירה. מ.ש.ל.

<u>דוגמא נוספת:</u> $L = \{ w \in \{0,1\}^* \mid$ מס' ה-1-ים ב-w שווה למס' ה-0-ים $\}$

L אינה רגולרית. אכן, נניח בשלילה שיש לה אוטומט בגודל k. נתבונן במילה $0^k 1^k \in L$. לפי למת השאיבה הקטע של 0-ים מכיל קטע ניפוח; אבל כל ניפוח/שאיבה מוציא אותנו מהשפה. ∎

אוטומטים (סופיים) לא דטרמיניסטיים (NFA)

למדנו תכונות שונות של שפות רגולריות, ודרכי הוכחה ששפה לא רגולרית. אבל, בכל זאת, מתמודד עם בעיית בניה של אוטומט עבור שפה (רגולרית) L? המושג של אוט' סופי לא דטרמיניסטי מקל את המלאכה הנ"ל בהרבה.

הגדרה: אוטומט סופי לא דט' מוגדר באופן דומה לזה שהגדרנו אוטומט דטרמיניסטי, עם הבדלים חשובים הבאים:

⊕ פונקצית מעברים δ: עבור כל זוג (q,σ)

* יכולה להיות לא מוגדרת;

* יכולה להיות לא חד-ערכית, לדוגמא $\delta(q,\sigma)=p_1$ וגם $\delta(q,\sigma)=p_2$

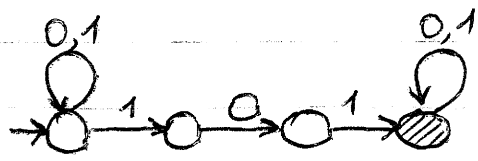
* בנוסף לזה, יתכנו מעברי-ε (מעברים ספונטניים ממצב למצב, ללא קלט): $\delta(q,\varepsilon)=p$.

⊕ הקבלה: מחרוזת $w\in\Sigma^*$ מתקבלת אם קיים מסלול המקבל אותה (כולל מעברי-ε). שים לב כי לאוט' לא דט', להבדיל מאוט' דט', יתכנו הרבה מסלולי חישוב, וגם יתכן כי אין אף מסלול חישוב.

שים לב: כל DFA הוא, לפי הגדרה, גם NFA

(עבור NFA אין טעם לדבר על זיהוי שפה, אלא רק על קבלת שפה).

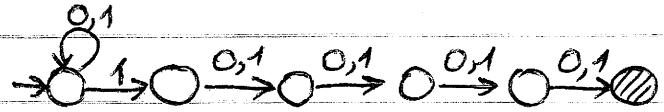
⊛ **דוגמא:** האוטומט הלא דט' מקבל שפה



$\{0,1\}^* \supseteq L=\{$מחרוזות המכילות רצף "101"$\}$

(15)

⊛ דוגמא נוספת: שפה $L = \{$ 1 לא הביט ה-5 מהסוף $\} \subseteq \{0,1\}^*$ מתקבלת ע"י



רק 6 מצבים!

כמובן, NFA לא ניתן ליישום מעשי ישיר. הערך של המודל הזה הוא בעובדה שלמרות הגמישות הרבה שבו, והקלות היחסית בה ניתן לתכנת בו, הוא עדיין מקבל רק שפות רגולריות!

⊛ **משפט:** יהי N אוט' לא דט'; אזי ק"ים (וניתן לבניה יעילה) אוטומט דטרמיניסטי A כך ש- $L_N = L_A$.

הוכחה: בניה:

N	מצבים	A
$Q=\{q_1,\dots,q_r\}$	$\dashrightarrow$	$\{K_i\}_{i=1}^{2^r}$ המתאימים לכל תת-קבוצות של Q

מצב התחלתי ב-A:

$K = \{$המצב ההתחלתי ב-N$\}$

מצבים מקבלים ב-A:

$\{K \mid K \cap F_N \neq \emptyset\}$ וגם המצב ההתחלתי החדש, אם מהמצב ההתחלתי ב-N ניתן להגיע ל-F ע"י מעברי ε.

כלו' הקבוצות המכילות מצב מסיים של N.

פונקצית המעבר

$K_i \xrightarrow{\sigma}$ קבוצת המצבים ב-Q שאליהם ניתן להגיע מ-$q \in K_i$ ע"י σ + מעברי ε.

אם אין כאלה בכלל:

$K_i \xrightarrow{\sigma} \emptyset$

יתפקד כבור.

מעצם הגדרת A ברור כי לכל $w \in \Sigma^*$

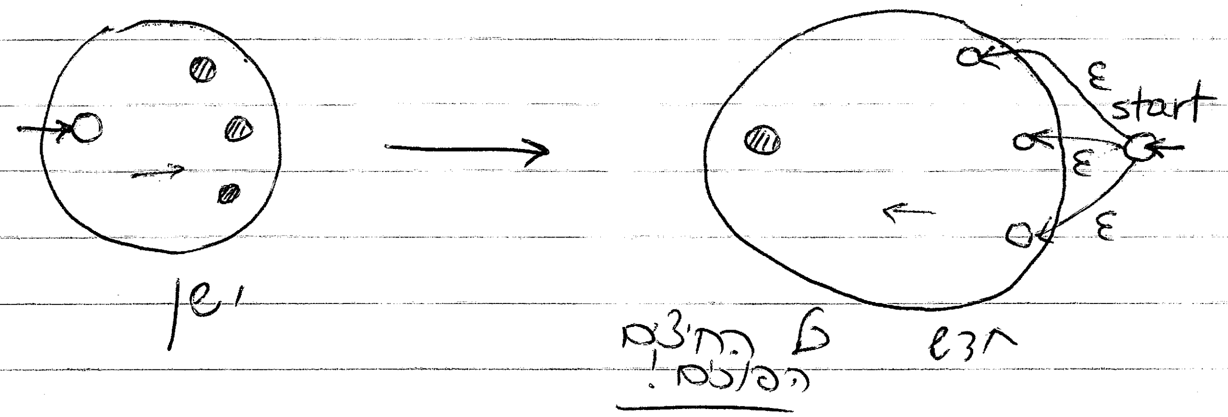
הרצה של A על w מסתיימת במצב K כך ש: $q \in K$ $\Longleftrightarrow$ הרצה של N על w יכולה להסתיים במצב q

לכן, יש מסלול מקבל של w ב N $\Leftrightarrow$ w מתקבל ב A, ובמילים אחרות, $L_A = L_N$. מ.ש.ל.

דוגמאות לשימוש:

* אם L רגולרית, אזי גם L^R (שפת ה reverse) רגולרית.

בניה: אותם המצבים + מצב אחד חדש (start).



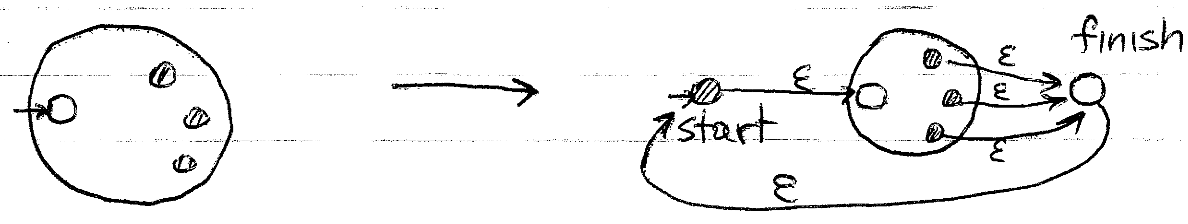
$\delta(p,\sigma) = q \Longrightarrow \delta'(q,\sigma) = p$

---

* אם L רגולרית, אז גם L^* רגולרית.

ישן:

חדש:



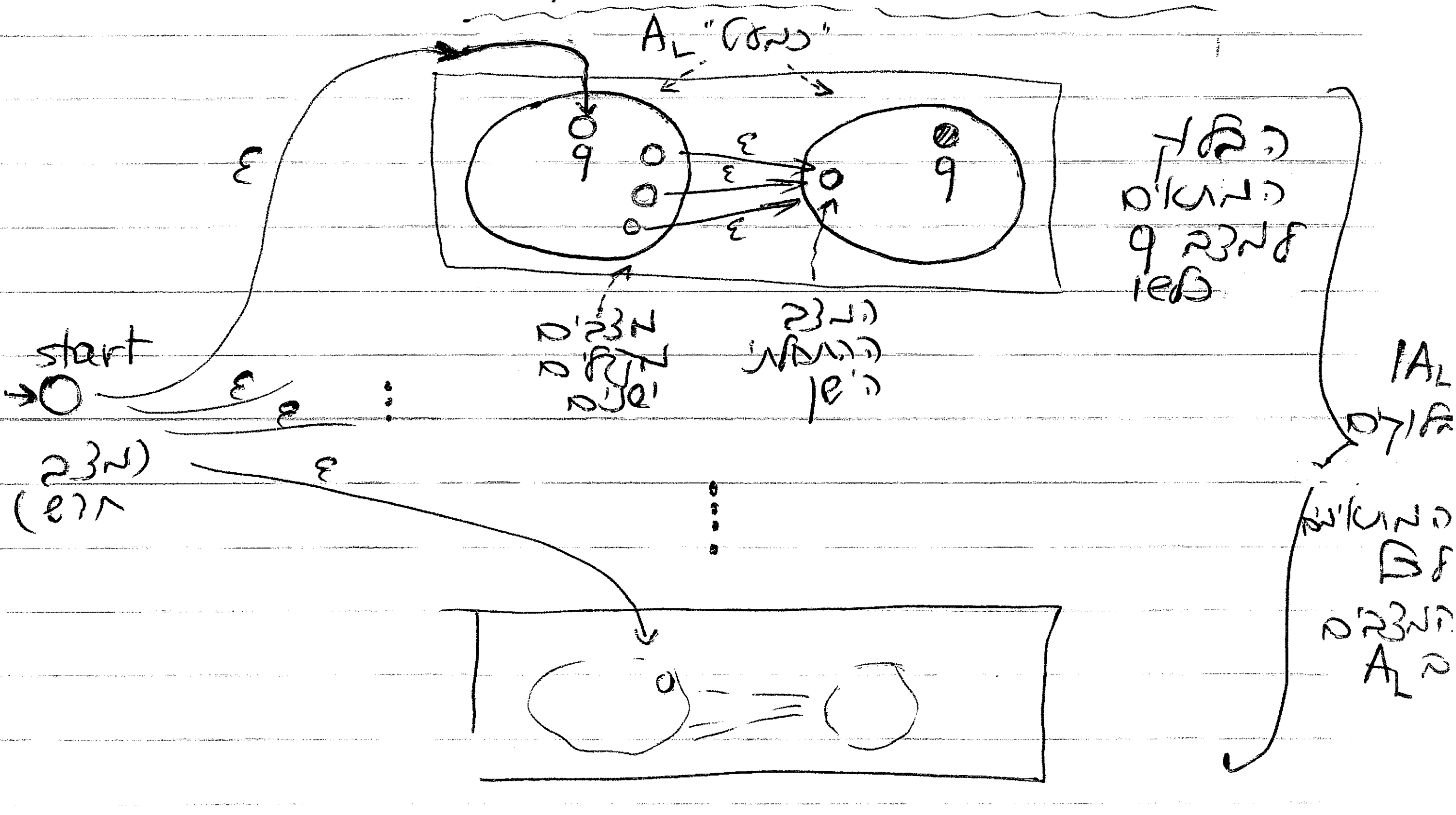
* דוגמא מסובכת יותר:

אם L רגולרית, אז גם L^{shift} כנ"ל, כאשר L^{shift} מכילה את כל המחרוזות המתקבלות מהמחרוזות של L ע"י shift ציקלי כלשהו. במילים אחרות,

$$L^{shift} = \{w_2 \cdot w_1 \mid w_1 \cdot w_2 \in L\} \supseteq L.$$

הבניה: האוטומט הלא דט' "ינחש" את המצב באוטומט המקורי A_L שבו $w = w_1 \cdot w_2$ המקורית "נשברה" והורכבה מחדש ל $w' = w_2 \cdot w_1$, ויקח את האיחוד של כל המקרים האפשריים (=מס' המצבים ב A המקורי). שים לב: לא ניתן לנחש את ה offset של השבירה: אוטומט סופי לא יכול לספור עד ∞...

התמונה של האוט' הלא דט':



שפות רגולריות וביטויים רגולריים

ביטוי רגולרי מעל א"ב Σ הוא ביטוי מהסוג

$$((a+b)^* \cdot (a\cdot b+b\cdot a^*)^* + aba^*b)^*$$

(ε, $\emptyset$ האותיות של Σ הן ביטויים רגולריים, וגם אם R_1, R_2 רגולריים, אזי גם R_1+R_2, $R_1\cdot R_2$, R_1^* הם כאלה).

+ מתארת פעולת האיחוד,
· — " — השרשור,
* — " — השרשור העצמי עם עצמו.

כך, $\emptyset$ ביטוי רגולרי מתאר שפה.

(*) **משפט** L רגולרית אם"ם יש ביטוי רגולרי המתאר אותה.

הוכחה (קונסטרוקטיבית):

כיוון אחד קל: אנחנו כבר יודעים שאם L_1, L_2 שפות רגולריות, אזי כך גם $L_1\cdot L_2$, $L_1\cup L_2$, L_1^*. כמו שהשפות $L_\sigma=\{\sigma\}$, $\sigma\in\Sigma$, הינן רגולריות, מ.ש.ל.

כיוון שני קשה יותר. עלינו להוכיח (לבנות), בהינתן אוטומט סופי המקבל את L, כי יש ביטוי רגולרי המתאר אותה.

שיטת הוכחה:

נתחיל עם האוטומט המקבל את L. כל עוד מס' המצבים גדול מ-2, ניצור אוטומט מוכלל שקול, בעל מצב אחד פחות. בסוף, נקבל את הביטוי הרגולרי מהאוטומט המוכלל בעל 2 מצבים.

+ אוטומט מוכלל:

מצב התחלתי אחד, מצב סופי אחד, לכל זוג מצבים $j \leftarrow i$ מתאימה שפה L_{ij}. המילה w מתקבלת ע"י אוטומט כזה אם אפשר להציג אותה

$L_{ij} \neq L_{ji}$!

$$w = \underset{\in L_{i_0 i_1}}{w_1} \cdot \underset{\in L_{i_1 i_2}}{w_2} \cdot \underset{\in L_{i_2 i_3}}{w_3} \cdot \ldots \cdot \underset{\in L_{i_{r-1} i_r}}{w_r}$$?

$$q_{start} = q_{i_0} \rightarrow q_{i_1} \rightarrow q_{i_2} \rightarrow q_{i_3} \rightarrow \cdots \rightarrow q_{i_{r-1}} \rightarrow q_{i_r} = q_{finish}$$

(התחלתי) (מקבל)

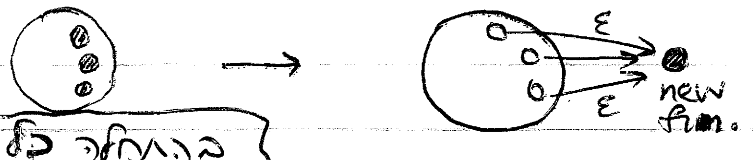
במילים אחרות, אוטומט מוכלל הוא לא דט', ובמקום לעשות צעד של אותיות או ε, כמו דטרמ', הוא אפשר צעד של מילה; אם הוא במצב q_i, ומגיעה מילה $w \in L_{ij}$, הוא יכול לעבור למצב q_j. מילה w מתקבלת אם קיים חישוב יחיד עליה, המתחיל ב q_{start} ומסתיים ב q_{finish}.

DFA (למשל)

א. נתחיל מהאוטומט המקבל את L, ונהפוך אותו למוכלל.

אם $\delta(q_i,\sigma) = q_j$, נגדיר $L_{ij} = \{\sigma\}$, ובכל מקרה אחר $L_{ij} = \emptyset$.

אם יש הרבה מצבים מס"מים, ניצור מצב חדש, ונקשר את המקבלים הישנים אליו במעבר ε.



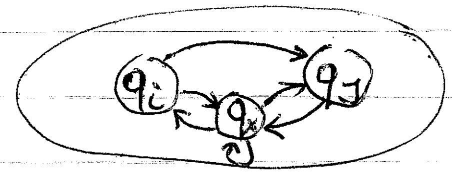
בהתחלה כל L_{ij}-ים ניתנים לתיאור ע"י ביטויים!

$q_i \notin F \rightarrow L_{i,new} = \emptyset$

$q_i \in F \rightarrow L_{i,new} = \varepsilon$

ב. איטרציה עיקרית: בהינתן אוטומט מוכלל המקבל את L, ויש בו $k>2$ מצבים, נבנה אוטומט מוכלל שקול, בעל $k-1$ מצבים.

הבניה: נבחר מצב $q_x \neq q_{start}, q_{finish}$. כיון שכל חישוב חוקי, אם הוא נכנס ל-q, ימשיך לצאת מ-q, נוכל להסיר באופציה את המעבר הזה:



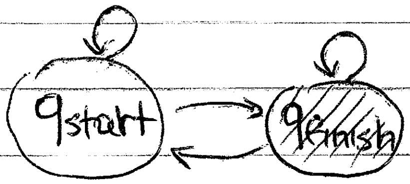
$\forall i,j \ (\neq x)$

$\boxed{\text{גם } i=j\,!}$ $\quad L_{i,j} \leftarrow L_{i,j} \cup L_{i,x} \cdot L^*_{x,x} \cdot L_{x,j}$

האוטומט החדש מכיל מצב אחד פחות (q_x), ואין זה קשה לבדוק כי הוא מחשב את אותה השפה L.

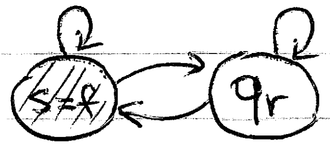
! שים לב: אם כל $L_{i,j}$-ים הישנים נתונים לתיאור ע"י ביטויים רגולריים, ע"פ הבניה שלנו, גם $L_{i,j}$-ים החדשים הם כאלה!

ג. סיום:



$\longrightarrow \quad L = \left(L^*_{ss} L_{sf} L^*_{ff} L_{fs}\right)^* \cdot L_{sf} \cdot L^*_{ff}$

או



$\dashrightarrow \quad L = \left(L^*_{ss} L_{sr} L^*_{rr} L_{rs}\right)^*$

קיבלנו הצגה של L המקורית כביטוי רגולרי!

מ.ש.ל.

דוגמא:

$S \to N \sqcup V \sqcup N$

$A \to asher \sqcup V \sqcup N / gadol / adom / yaffe / \varepsilon$

$N \to N \sqcup A / yeled / chatul / dag / kelev$

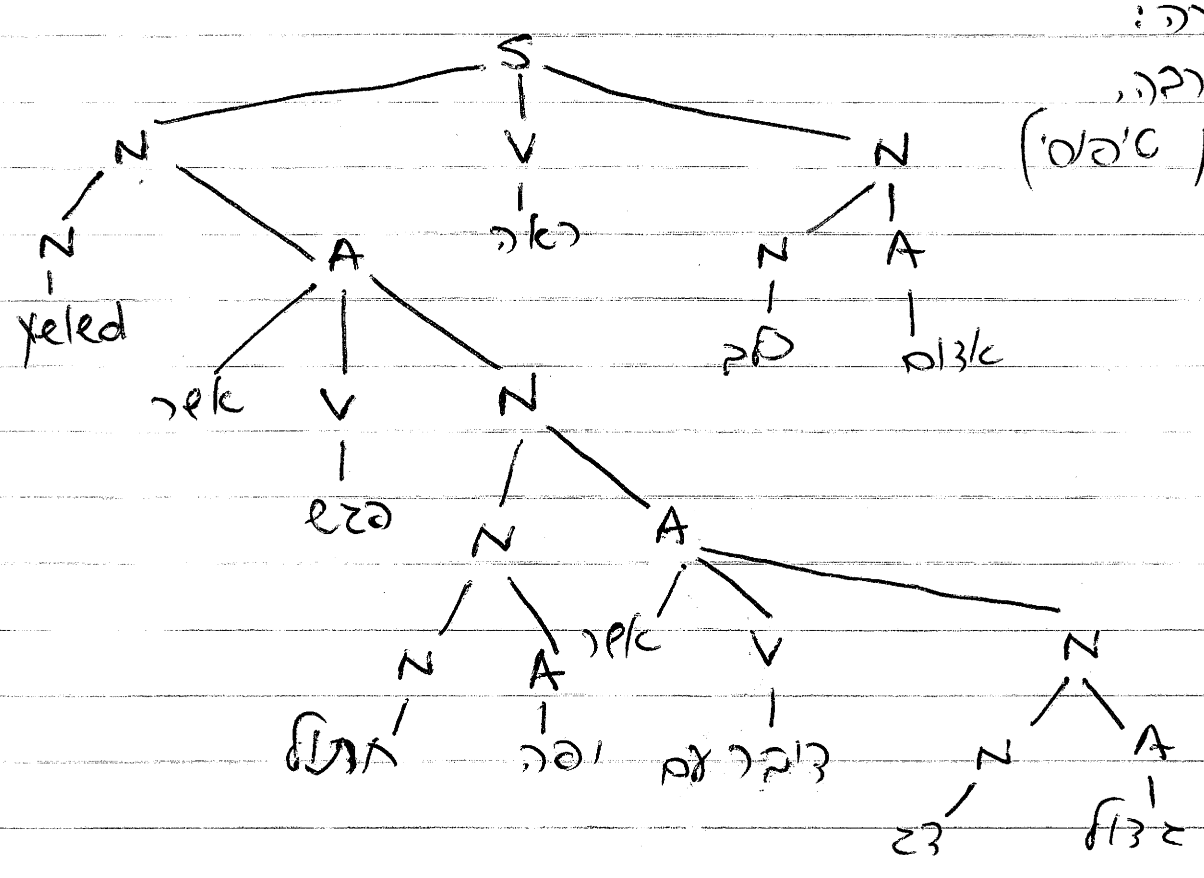
$V \to raa / pagash / dibber \sqcup im$

- מלה בשפה:

ילד אשר פגש חתול יפה אשר דיבר עם דג גדול ראה כלב אדום.

- גזירה:

יש הרבה,
באופן טיפוסי)



---

* דוגמא 1:

שפה:

$L = \{0^i 1^i\}_{i=0}^{\infty}$

דקדוק:

$S \to 0S1 / \varepsilon$

פרק ב'

שפות חסרות הקשר

עבור משפות רגולריות ואוטומט סופי למחלקת שפות רחבה יותר, המתאימה למודל חישובי חזק יותר (אוטומט מחסנית).

שפות חסרות הקשר היא מחלקה טבעית של שפות, הבאה מניסיון לתת מודל הולם לדקדוקים של שפות אנושיות. הן הוגדרו לראשונה ע"י הבלשן נועם חומסקי. מאוחר יותר נמצא מודל חישובי המתאים להם, ופותחו כלים לניתוח של שפות כאלה.

(*) דקדוק חסר הקשר:

- סימבולים: משתנים V, טרמינלים Σ.
- אחד המשתנים (בד"כ S) נקרא התחלתי.
- חוקי גזירה: נראים כ: $v \longrightarrow w$ כאשר $v \in V$, $w \in \{\Sigma \cup V\}^*$. יש מספר סופי של חוקים כאלה.

(*) שפה חסרת הקשר:

(עפ"י דק', נגזרת לגזירה)

שפה מעל Σ, הנגזרת מדקדוק ח"ה. נגזרת: מתחילים מ S, ובכל שלב בוחרים משתנה מסוים V_i, ומציבים במקומו w_i (בתנאי שיש חוק $V_i \to w_i$). עוצרים כאשר אין עוד משתנים.

$\Sigma=\{a,b\}$

* דוגמא 2:

שפה: $\{ww^R\}$. $\longrightarrow$ דקדוק:

$S \to aSa / bSb / \varepsilon$

* דוגמא 3:

שפה: $\{wu \mid |w|=|u|, w^R \neq u\}$ $\longrightarrow$ דקדוק:

$S \to aSa / bSb / aRb / bRa$

$R \to aRa / aRb / bRa / bRb / \varepsilon$.

* דוגמא 4:

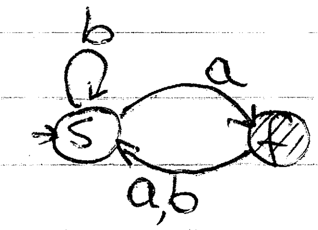
אם L רגולרית, היא גם ח"ה. אכן, יהי A האוטומט המקבל אותה; נניח לנוחות כי שם המצב ההתחלתי הוא S.

- משתנים: שמות המצבים;
- חוקי גזירה:

• אם $\delta(q,\sigma)=p$ $\longleftarrow$ $q \to \sigma p$

• אם $q \in F$ $\longleftarrow$ $q \to \varepsilon$.

לדוגמא,



$S \to bS / aF$

$F \to aS / bS / \varepsilon$.

(23)

$\Sigma=\{a,b\}$

* דוגמא 5

שפה: $\{w \mid \#a = \#b\}$

דקדוק:

$S \to SS / aSb / bSa / \varepsilon$

דקדוק אחר:

$S \to AB / BA / \varepsilon$

$A \to ABA / AAB / a$

$B \to BBA / BAB / b$

* דוגמא 6:

שפה: $\{w \mid \#a \geq \#b\}$

דקדוק:

$S \to a / aSb / bSa / SS / \varepsilon$

* דוגמא 7:

$\Sigma=\{a,b,c\}$, $L=\{a^i c^j b^k \mid i+k=j\}$

$S \to XY$

$X \to aXc \mid \varepsilon$

$Y \to cYb \mid \varepsilon$

$$\boxed{\begin{array}{l} S \to AB \mid BA \\ A \to aAa \mid aAb \mid bAa \mid bAb \mid a \\ B \to aBa \mid aBb \mid bBa \mid bBb \mid b \end{array}}$$

הכיוון: צריך לבנות

$\underbrace{\square}_{i} * \underbrace{\square}_{i+j} * \underbrace{\square}_{j}$ מצב ?

* דוגמא 8:

$\Sigma=\{a,b\}$, אלפ

$L=\{wu \mid |u|=|w|, u \neq w\}$

אוטומט מחסנית

* המודל החישובי המתאים לשפות ח"ה הינו אוטומט מחסנית.

זה אוטומט של רג', בעל זכרון פנימי סופי, וזכרון חיצוני המאורגן בצורת מחסנית, כ"א בכל רגע ורגע יש גישה לאיבר העליון של המחסנית בלבד. ניתן למחוק את האיבר הנ"ל (POP), להוסיף איבר חדש מעליו (PUSH), או לרשום משהו במקומו.
מחרוזת w מתקבלת ע"י אוטומט כנ"ל, אם קיים עליה חישוב חוקי, המסתיים במצב פנים מקבל, ובמחסנית ריקה.

פורמלית:

Q: מצבים (q_0 התחלתי, F: מקבלים);

Γ: א"ב המחסנית ($\$$ סימן מיוחד, הנמצא בתחתית המחסנית)

Σ: א"ב הקלט;

פונקצית מעבר: $\delta: Q \times \Gamma \times \Sigma \cup \{\varepsilon\} \to Q \times \Gamma \cup \{remove\}$

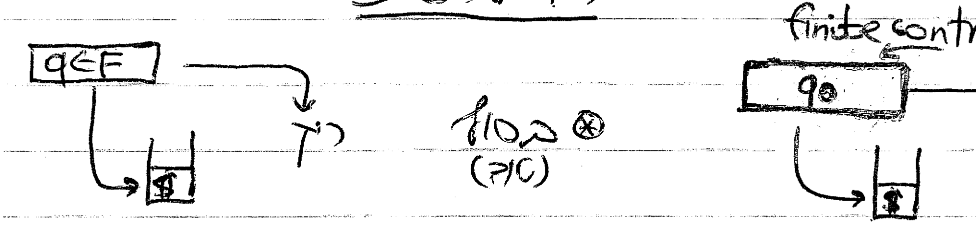
PUSH

רק ערכית

חישוב חוקי:

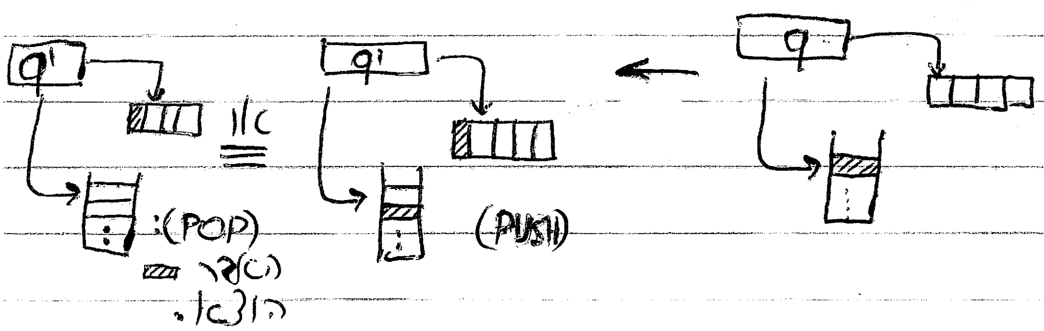
* בהתחלה

* בסוף (סוף)



אסור
או - אפשר
למחוק את $!

* מעבר:



יש גם
מעברי ε!

כדי לתכנת באוטומט מחסנית, נוח להשתמש בהגדרה שקולה, אך גמישה יותר

$$\delta: Q \times \Gamma \times \Sigma \cup \{\varepsilon\} \longrightarrow Q \times \Gamma^*$$

כאשר במקום הסימן העליון של המחסנית מותר לכתוב מילה שלמה במקום סימן אחד ($remove \equiv \varepsilon$).

(23)

* דוגמא: שפת הסוגריים המאוזנים $L \subseteq \{(,)\}^*$

(למשל, $(()())((())) \in L$).

$Q = \{q, f\}$ q המתחיל והמקבל

$\Gamma = \{\$, (\ \}$

$q, (\xrightarrow{)} q, POP$;

$q, (\xrightarrow{(} q, PUSH(()$;

$q, \$ \xrightarrow{(} q, PUSH(()$; $q, \$ \xrightarrow{)} f, PUSH(()$;

$f, * \xrightarrow{\sigma} f, PUSH(\sigma)$;

האוטומט במקרה זה דטרמיניסטי

בשפה שלנו:

```
P:
state ← q;
while (INPUT ≠ Ø  ∧  state ≠ f)
{   read(I);
    if (I = ")") {if (STACK = "(") POP;  else state ← f}
        else  PUSH("(")
}
if (state = q) and STACK = "$"  accept.
```

ה) הצגה בשפה שלנו נוחה יותר, ומדויקת אף
(עוד היא אכן מתאימה לאוטומט מחסנית!).

* שים לב: בגלל א'-ג', תהיה שם פקודה חדשה
"נחש את הפעולה המתאימה מבין ...".

שקילות בין השפה המתקבלת ע"י אוטומט מחסנית לבין שפה ח"ה

* כיוון דקדוק ← אוטומט:

בהינתן דקדוק ח"ה G עלינו לבנות אוטומט מחסנית המקבל בדיוק את השפה המוגדרת ע"י G.

רעיון הבניה: האוטומט פשוט ינחש את כל הגזירות השמאליות בעצמו, וישווה את המילה ב-Σ^* שהגיע אליה עם ה INPUT. אם צדק, נקבל.

A_G:

$Q: \{q_0, q\}$ q_0: מתחיל, q: מקבל

$\Gamma: \{\$\} \cup \Sigma \cup V$

משתנים וטרמינלים של הדקדוק G

פונקצית מעבר:

(* התחלה *)	$q_0, \$ \xrightarrow{\varepsilon} q, S\$$	(PUSH(S) בכה) ←
(* לכל גזירה $V \to w$ *)	$q, V \xrightarrow{\varepsilon} q, w$	
(* לכל $\sigma \in \Sigma$ *)	$q, \sigma \xrightarrow{\sigma} q, \varepsilon$	(POP בכה) ←

הערה: הבדיקה של שיוויון מתבצעת ONLINE; באופן כזה נדאג לכך שב top המחסנית יהיו משתנים ולא טרמינלים.

הגזירה היא השמאלית ביותר המתאימה לסריקה של עץ הגזירה באופן הבא: מתחילים מהשורש; באופן רקורסיבי האב זוכר את בניו משמאל לימין, ומטפל בהם משמאל לימין.

* בניית דקדוק שקול לאוטומט מחסנית

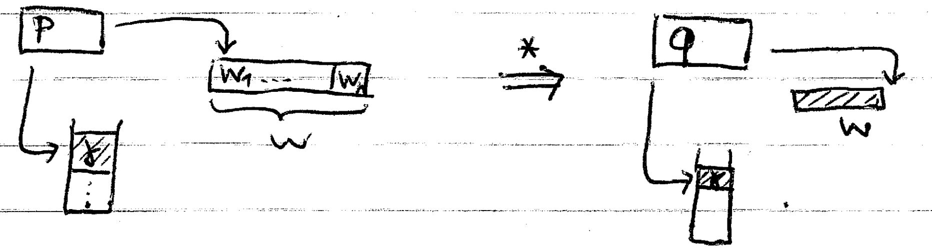
$G_A \leftarrow A$

יהי $(Q,\Gamma,\Sigma,\delta,q_0,F)$ התיאור הפורמלי של האוטומט A;

(אנו משתמשים בהגדרה המקורית, בו מותר למחוק סימן עליון, או לעשות PUSH לסימן חדש, וזהו).

לכל $p,q\in Q$, $\gamma\in\Gamma$. נגדיר שפה $L^{\gamma}_{p\to q}\subseteq\Sigma^*$

כאוסף כל המחרוזות w כך ש- ד"מ חישוב



אשר לעולם לא עשה POP ל-γ העליון.

מתקיים:

$$\boxed{L_A=\bigcup_{f\in F}L^{\$}_{q_0\to f}}$$

וגם, לכל $L^{\gamma}_{p\to q}$

* אם $(p,\gamma)\xrightarrow{\sigma_1}(p',PUSH(\gamma_1))\wedge(p'',\gamma_1)\xrightarrow{\sigma_2}(q,POP)$,

$$L^{\gamma}_{p\to q}\supseteq\sigma_1\cdot L^{\gamma_1}_{p'\to p''}\sigma_2;$$

** לכל $p'\in Q$

$$L^{\gamma}_{p\to q}\supseteq L^{\gamma}_{p\to p'}\cdot L^{\gamma}_{p'\to q}.$$

*** $L^{\gamma}_{p\to p}\supseteq\{\varepsilon\}$

יתר על כן, אם ניקח את האיחוד של כל אלה, נקבל בדיוק $L^{\gamma}_{p\to q}$.

זה מביא אותנו להגדרה הבאה של דקדוק ח"ה G_A:

המשתנים: $L^{\gamma}_{p\to q}$, לכל $p,q\in Q,\ \gamma\in\Gamma$
S !.

חוקי גזירה:

שים לב: $L^{\gamma}_{p\to q}$ הפכו אותם למשתנים! הקשר עם השפה הוא שהמשתנה $L^{\gamma}_{p\to q}$ יוצר בדיוק שפה $L^{\gamma}_{p\to q}$

(1.) $\boxed{S \to L^{\$}_{q_0\to f_1} \mid \dots \mid L^{\$}_{q_0\to f_r}}$ $\{f_1,\dots f_r\}=F\subseteq Q$

(2.) $\boxed{L^{\gamma}_{p\to q} \to \sigma_1 L^{\gamma_1}_{p'\to p''}\sigma_2}$

לכל $p,q,p',p''\in Q$, $\gamma\in\Gamma$, $\sigma_1,\sigma_2\in\Sigma\cup\{\varepsilon\}$ כך ש:

$(p,\gamma)\xrightarrow{\sigma_1}(p',\ PUSH(\gamma_1))$ וגם $(p'',\gamma_1)\xrightarrow{\sigma_2}(q,POP)$

(3) $\boxed{L^{\gamma}_{p\to q} \to L^{\gamma}_{p\to p'}\cdot L^{\gamma}_{p'\to q}}$ $\forall p,q,p'\in Q$, $\gamma\in\Gamma$

(4) $\boxed{L^{\gamma}_{p\to p} \to \varepsilon}$ $\forall p\in Q,\ \gamma\in\Gamma$

(*) דוגמא: (נושפה) $\{a,b\}^* \supseteq L=\{w \mid \#a=\#b\}$

אוטומט:
$\Gamma=\{a,b,\$\}$, $Q=\{q\}$ המצב היחיד

$q,\$ \xrightarrow{a,b} q,\ PUSH$
$q,a \xrightarrow{b} q,\ POP$
$q,b \xrightarrow{a} q,\ POP$
$q,a \xrightarrow{a} q,\ PUSH$
$q,b \xrightarrow{b} q,\ PUSH$

דקדוק:

$S\to L^{\$}_{q\to q}$
$L^{\$}_{q\to q}\to\varepsilon \mid L^{\$}_{q\to q}\cdot L^{\$}_{q\to q} \mid a L^{a}_{q\to q} b \mid b L^{b}_{q\to q} a$
$L^{a}_{q\to q}\to\varepsilon \mid L^{a}_{q\to q}\cdot L^{a}_{q\to q} \mid$ —״— $\mid$ —״—
$L^{b}_{q\to q}\to\varepsilon \mid L^{b}_{q\to q}\cdot L^{b}_{q\to q} \mid$ —״— $\mid$ —״—

לסיכום, תיארנו בניה מפורשת הבונה אוטומט מחסנית שקול לדקדוק ח"ה נתון, ובניה מפורשת הבונה דקדוק ח"ה שקול לאוטומט מחסנית נתון. מעתה והלאה, לצורך דיון בשפות ח"ה, נוכל להשתמש בכל אחד מהמודלים הנ"ל, לפי נוחיותנו.

* דוגמא: שימוש באוטומט

$$L = \{ w_1 \square w_2 \mid w_1, w_2 \in \{a,b\}^*, \; w_1 \neq w_2 \}$$

(* אין דרישה לאורך שווה *)

הרעיון: א. נכניס את אברי w_1 למחסנית עד למקום ה-k (אותו ננחש) בו $w_1(k) \neq w_2(k)$ (המקרה הטיפוסי; המקרים הלא טיפוסיים - w_1 הוא רישא ממש של w_2, וכו' - יטופלו תוך כדי).

ב. לא נשנה את מצב המחסנית עד אשר נגלה "$\square$".

ג. נזכור את האות בראש המחסנית; נבצע קלט במקביל לריקון המחסנית (כ"א, אנו מיישמים COUNTER) - בדרך זאת נאתר את $w_2(k)$.

ד. אם האות שזכרנו שונה מ $w_2(k)$, accept.

אוטומט המחסנית - פורמט תוכנית:

זכרון בנוי: $q_{start}, q^1_\square, q^1_a, q^2_a, q^1_b, q^2_b, q_{accept}, q^2_\square$

```
(*) state ← qstart;
    while (state = qstart)
    { READ(INPUT); guess: • state ← q¹_INPUT
                            OR
                          • PUSH(INPUT) }
    while (INPUT ≠ □) READ(INPUT);
    state ← q²_*   (* q¹_{a,b,□} → q²_{a,b,□} *);
    while (STACK ≠ $ and INPUT = ∅)
    {read(INPUT);  POP}
    if ((state = q²_a ∧ INPUT = b) ∨
        (state = q²_b ∧ INPUT = a) ∨
        (state = q²_□ ∧ INPUT ≠ ∅) ∨
        (state ≠ q²_□ ∧ INPUT = ∅))
    {state ← qaccept; empty(STACK);
     accept}.
```

30 ⊛ **דוגמא:** אם L שפה ח"ה המתקבלת ע"י אוטומט מחסנית דטרמיניסטי, אזי $\bar{L}$ גם ח"ה.

אכן, נשתמש באותו האוטומט כמו ל-L; אם בסיום הקלט האוטומט נמצא במצב לא מקבל, נריקן את המחסנית ונקבל.

(שים לב: בהגדרה של אוטומט לא דט', השיקול הנ"ל לא עובד, כיוון שיתכנו המון חישובים שונים, וכדי לקבוע כי כולם לא מקבלים, יש צורך לעבור על כולם... אכן, גם הטענה לא מתקיימת: יש שפות ח"ה, שהמשלים שלהם אינו כנ"ל.)

דוגמא לשפה כזאת: $\{0^n1^n\}^c$

⊛ **דוגמא:** אם L שפה ח"ה, ו-R שפה רגולרית, אזי $L \cap R$ ח"ה.

הרעיון מאוד פשוט: נריץ את A_L ו-A_R במקביל על הקלט: כיוון שהריצה של A_R אינה דורשת שום זכרון חיצוני, ולכן אפשר לבצע אותה בראש הקורא (FINITE CONTROL), הדבר אפשרי. פורמלית:

A_L:	A_R (דט'!):		$A_L \times A_R$
Q_L, Γ, δ_L	Q_R, δ_R	$\longrightarrow$	$Q_L \times Q_R$, Γ,
q_0^L, F_L	q_0^R, F_R		(q_0^L, q_0^R), $F_L \times F_R$

פונקציית מעבר:

⊛ $\left\{ \begin{array}{l} q^L, \gamma \xrightarrow{\sigma} p^L, * \\ q^R \xrightarrow{\sigma} p^R \end{array} \right\} \Rightarrow (q^L, q^R), \gamma \to (p^L, p^R), *$

אם (בישן) $\Rightarrow$ אזי (בחדש)

⊛⊛ $\{ q^L, \gamma \xrightarrow{\varepsilon} p^L, * \} \Rightarrow \forall q^R,\ (q^L, q^R) \xrightarrow{\varepsilon} (p^L, q^R), *$

שים לב: אם L_1 ו-L_2 ח"ה, לא ניתן לעשות סימולציה כזאת, כיוון שנדרשות שתי מחסניות.

(31)

⊛ <u>דוגמא</u>: תהי $PREFIX(L)$ השפה המוגדרת באופן הבא:

$$PREFIX(L) = \{u \in \Sigma^* \mid \exists w \in L, \text{ ש-}u\text{ היא רישא של } w\}$$

<u>טענה</u>: אם L ח"ה, אזי גם $PREFIX(L)$.
ניתן שתי הוכחות.

① יהי A_L אוטומט מחסנית המקבל את L; נעבור בו כדי לבנות אוטומט $\tilde{A}$ המקבל את $PREFIX(L)$.
הרעיון: $\tilde{A}$ יסמלץ את A_L עד סוף הקלט, ומרגע זה יסמלץ קלט וירטואלי; אם יש אפשרות לנחש המשך של u כדי לקבל $w \in L$, $\tilde{A}$ יעשה זאת.

<u>פורמלית</u>:

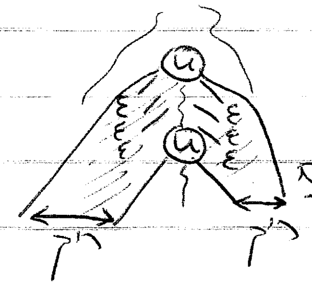
A_L	$\tilde{A}$
Q, Γ, Σ	Q, Q', Γ, Σ
q_0, F	q_0, F'
$(q,\gamma) \xrightarrow{\sigma} p, *$	$(q,\gamma) \xrightarrow{\sigma} p, *$
$(\forall q,\gamma,\sigma,p)$	$(q',\gamma) \xrightarrow{\varepsilon} p', *$
	וגם $\forall q \in Q,\ (q,\gamma) \xrightarrow{\varepsilon} q'$, (אין שינוי במחסנית STACK)

② דרך דקדוקית:
לכל משתנה $V \in G$ (כולל S) נגדיר משתנה חדש V' האמור להתאים לשפת הרישות של השפה הנגזרת מ-V ב-G.

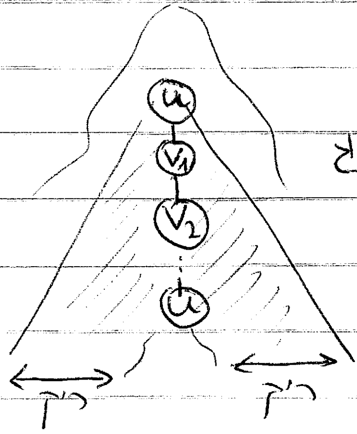
כך בדיוק עובד מנגנון השאיבה/ניפוח.

יש שני מקרים כאשר לא נקבל שום דבר מעניין:

1. ε-מעברים



2. מעברים מסוג $V_i \to V_j$



נראה תחילה כי בה"כ אפשר להמנע גם מ-1 וגם מ-2.

* 1. תהי L שפה ח"ה. אזי ד"ם ל-$L \setminus \{\varepsilon\}$ דקדוק $\tilde{G}$ ח"ה חסר מעברי ε.

ניגש להבנה. נתחיל מ-G, דקדוק ח"ה ל-L.

א. נאתר את המשתנים $X \in V$ כך ש-$X \xrightarrow{*} \varepsilon$.
אלגוריתמית, נאתר את המשתנים שגוזרים את ε בצעד אחד, אחרי זה את המשתנים הגוזרים בצעד אחד את המשתנים הנ"ל, וכו'.
(או לאתר דיוק, מילים המורכבות מהמשתנים הנ"ל בלבד)

ב.

G	$\tilde{G}$
$V \to \varepsilon$	אין
$V \to \dots X_1 \dots X_2 \dots X_k \dots$ $X_1, X_2 \dots X_k \xrightarrow{*} \varepsilon$	2^k כל החוקים המתקבלים ע"י הצבת ε-ים ב(חלק) מה-X_i-ים (מבלי ליצור $V \to \varepsilon$!)

G (ישן)	$\tilde{G}$ (חדש)
V, Σ	V, V', Σ
	כמו קודם $X \to X_1 \dots X_k$;
$X \to X_1 X_2 \dots X_i \dots X_k$	$X' \to X_1 \dots X_{i-1} X_i'$
$\forall X_i \in V$	וגם,
($\forall X, X_i \in V$, חוקי בגזירה)	$\forall X' \in V'$, $X' \to \varepsilon$

S' start

כדי להראות כי $\tilde{G}$ גוזר בדיוק PREFIX(L), מספיק להוכיח באינדוקציה על k (אורך המילה) כי לכל $X' \in V'$,

$$L_{X'}^{\le k} \text{ ב } \tilde{G} = \{ \text{הרישות באורך} \le k \text{ של } L_X \text{ ב } G \}$$

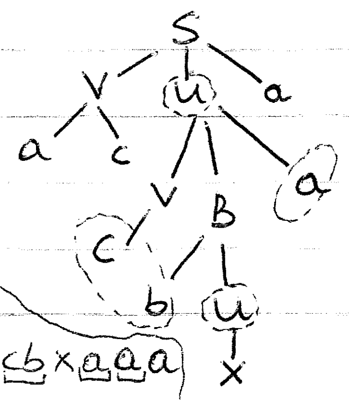
כל המילים באורך $\le k$ הניתנות לגזירה ב $\tilde{G}$ ממשתנה X'

כל המילים הניתנות לגזירה ממשתנה X ב G.

לא נכנס כאן לפרטים הפורמליים.

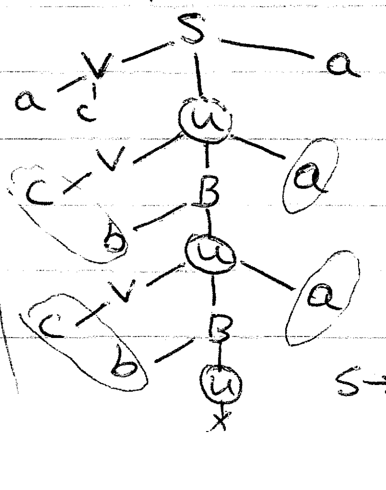
* למת השאיבה/ניפוח לשפות ח"ה

נניח כי הגזירה הבאה חוקית ב G:



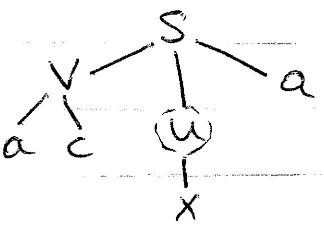
$S \to accbxaa$

אזי הגזירה הבאה גם חוקית



$S \to ac\underline{cb}\,\underline{cb}x\underline{a}\,\underline{a}a$

וגם כאלו:



$S \to acxa$

ובאופן כללי, $\forall k$,
$S \to ac(cb)^k x a^k a$

2. תהי L שפה ח"ה. אזי קיים ל-L דקדוק ח"ה $\tilde{G}$
חסר מעברי $V_i \to V_j$ (משתנה ← משתנה בודד).

וזאת הבניה. נתחיל מ-G, דקדוק ח"ה ל-L; נעבור ל-$\tilde{G}$, דקדוק ל-$L\setminus\{\varepsilon\}$ ללא מעברי ε. עתה, לכל זוג משתנים V_i, V_j ב-$\tilde{G}$, נבדוק האם יש גזירה $V_i \xRightarrow{*} V_j$.

אלגוריתמית, זאת בדיקת קשירות בגרף בו הקדקודים הם המשתנים, ויש $X \to Y$ אם יש גזירה כזאת בקדקוד

התרגום:

$\tilde{G}$	$\tilde{\tilde{G}}$
$V \to$ (לא משתנה בודד)	ללא שינוי
$V \to$ (משתנה בודד)	אין
	$\forall V_i \xRightarrow{*} V_j,\ V_j \to W$ (לא משתנה בודד!) נוסיף חוק חדש $V_i \to W$

למת השאיבה/ניפוח:

תהי L שפה ח"ה. נתבונן בדקדוק $\tilde{\tilde{G}}$ חסר מעברי ε, ומעברי $V_i \to V_j$ (יש כזה - הוכחנו). יהי n מס' המשתנים ב-G, יהי r האורך המקסימלי של חוק גזירה ב-$\tilde{\tilde{G}}$: $V \to \underbrace{W}_{\le r}$. אזי כל מלה $y \in L$ מאורך $> r^{n+1}$ ניתנת לפירוק בצורה

$$Y = u_1x_1u_2x_2u_3$$

כך ש:

* $x_1 \neq \varepsilon$ או $x_2 \neq \varepsilon$;

** $|x_1u_2x_2| < r^{n+1}$;

*** לכל $k \in \mathbb{N}\cup\{0\}$, $u_1x_1^ku_2x_2^ku_3 \in L$.