מ.מ.למדמ"ח $\sim$ משהו?

שחר פרץ

29 במאי 2024

1 דקדוק – תזכורות

תזכורת:

$$G = (V, \Sigma, R, S)$$

- V – משתנים
- Si – אלפבית
- R – כללי גזירה
- $S \in V$ – משתנה התחלה

דוג':

$$V = \{A, B\} \tag{1}$$

$$\Sigma = \{0, 1, \#\} \tag{2}$$

$$\begin{aligned} R\colon\ A \to 0A1 \iff & A \to 0A1 \mid B \\ A \to B \qquad\quad & B \to B \\ B \to B \end{aligned} \tag{3}$$

$$S = A \tag{4}$$

במקרה הזה:

$$L(G) = \{0^n\#1^n \mid n \in \mathbb{N}\}$$

לדוגמה $00\#11 \in L(G)$. תזכורת: אנו נתמקד אך ורק בדקדוקים חסרי הקשר – בכללי הגזירה, בצד שמאל מפיע רק משתנה אחד (וללא בתווים מ־Σ). לדוגמה, דקדוק שיכלול את הכלל $aV \to A$, יהיה לא חסר הקשר. (חסר הקשר = Context Sensitive ודקדוק לא חסר הקשר = Context Free Grammers).

שאלות ששאלנו בשיעור שעבר:

- האם $L(G_1) = L(G_2)$?
- האם $L(G_1) \subseteq L(G_2)$?
- האם יש יותר מעץ גזירה אחד לכל ילה ב־$L(G_1)$?
- האם $L(G_1) \cap L(G_2) \neq \emptyset$?

כל הדברים להלן, שקולים לבקשה לכתוב תוכנת מחשב המקבלת כקלט תוכנית מחשב, ולהחזיר אם התוכנית תיגמר בזמן סופי או לא. השאלה הזו בלתי אפשרית לפתרון סופי.

השאלות האלו קרויות **לא כריעות** – מלשון הכרעה, לא ניתן להכריע את התוצאה שלהן.

נרצה לענות בזמן טוב על שאלות יותר קלות: בהינתן דקדוק חסר הקשר G, ומילה w:

- האם $w \in L(G)$ (Recognition)?
- אם $w \in L(G)$, למצוא עץ גזירה כלשהו של w ב־G (Parsing).

2 פתרון הבעיות

2.1 CNF

על מנת לענות על השאלות הללו, נמיר דקדוק ל־Chomsky Normal Form, ובצורה זו קל יותר לענות על השאלות האלו.

הגדרה: דקדוק חסר הקשר הוא מצורת CNF אמ"מ כל כללי הגזירה בו הם מהצורה הבאה:

$$A \to a, \qquad A \in V,\ a \in \Sigma \tag{5}$$
$$A \to BC \quad A \in V, B, C \in V \setminus \{S\} \tag{6}$$
$$S \to \varepsilon \tag{7}$$

שימו לב שזו הגדרה הנכונה, בזו של השיעור שעבר נפלה טעות

טענה: כל דקדוק חסר הקשר ניתן לכתיבה והמרה בצורת CNF.

כדי "להוכיח", נראה כיצד נפתור כללי גזירה לא תקינים:

$$A \to BCD \quad \Longrightarrow \quad \begin{cases} A \to BE \\ E \to CD \end{cases} \tag{8}$$

$$A \to \varepsilon \quad \Longrightarrow \quad \begin{cases} B \to A \\ B \to a|aA \end{cases} \tag{9}$$

$$A \to aB \quad \Longrightarrow \quad \begin{cases} A \to A'B \\ A' \to a \end{cases} \tag{10}$$

נתרגם את בחלק (1) למעלה ל־CNF. שני הכללים הראשונים לא עונים להגדרה.

$$\begin{cases} A \to 0A1 \\ A \to B \\ B \to \# \end{cases} \quad \Longrightarrow \quad \begin{cases} C_0 \to 0 \\ C_1 \to 1 \\ A \to C_0AC_1 \\ A \to B \\ B \to \# \end{cases} \quad \Longrightarrow \quad \begin{cases} C_0, C_1 \to 0, 1 \\ A \to C_0A \\ A_1 \to AC_1 \\ A \to \# \end{cases} \tag{11}$$

2.1.1 סיכום

בצורת CNF עלולים להיות יותר משתנים ויותר כללי גזירה, ובכלל, הדקדוק עלול להיות פחות קריא לאנשים.
אבל, עבור המחשב, צורה זו מאוד נוחה.

2.2 אלגו' Cocke-Younger-Karzem – CYK

אלג' לזיהוי ופירוש של מילה בהינתן דקדוק חסר הקשר בצורת CNF.

רעיון (רקורסיבי): אם $G = (V, \Sigma, R, S)$ ונתונה המילה w, נאמר $w \in L(G)$ אמ"מ מתקימים אחד משלושת התנאים הבאים:

- $w = \varepsilon$ וקיים הכלל $S \to \varepsilon$.
- $w = x \in \Sigma$ וקיים הכלל $S \to x$.
- קיימת חלוקה $1 \le k \le |w| - 1$ של w לשתי מחרוזות לא ריקות $w[:k],\ w[k:]$ וקיים כלל $S \to XY$ כך שניתן לגזור את $w[:k]$ מ־X וניתן לגזור את $w[k:]$ מ־Y.

זהו לא פתרון יעיל, אך הוא עובד, וסופי, ונוכל ליעל אותו.

לצורך נוחות, נסמן מתשנים באותיות גדולות באנגלית ותווים בעזרת אותיות קטנות, ואת כללי הגזירה נייצג במילון עם משתנה לקבוצת הכללים שלו. לדוגמה, $A \to AB,\ A \to a,\ B \to b$ יהיה שקול ל-:

```
R = {A: {AB, a}, B: {b}}, Si = {A, B}, V = {a, b}, S = A
```

2.3 קוד

זה הזמן להתחיל לחרבש קוד:

```
def CYK(rules, start_var, w):                          # w, i, j)
    if len(w) < 2:                                     # j - i < 2
        return w in rules(start_var)                   # w[i:j]
    for k in range(1, len(w)):                         # range(i + 1, j)
        for rule in rules[start_var]:
            if len(rules) == 2:
                if CYK(rules, reuls[0], w[:k]) and \   # w, i, k)
                   CYK(rules, rules[1], w[k:]):        # w, k, j)
                       return True

    return False
```

דוג':

$$V = \{S, A < B < C\} \tag{12}$$

$$\Sigma = \{a, b\} \tag{13}$$

$$\begin{aligned} R\colon\ S &\to AB|BC \\ A &\to BA|a \\ B &\to CC|b \\ C &\to AB|a \end{aligned} \tag{14}$$

$$w = baaba \tag{15}$$

נמצא שכבר בקריאה הראשונה לרקורסיה, נעשה 16 קריאות רקורסיביות עוד בקריאה הראשונה לפונקציה. לא אפרט איך זה עובד למאות שהמורה פירט את זה על הלוח כי זה נראה לי מיותר.

יש כאן פעולה מאוד מיותרת ויקרה – slicing. נוכל לשנות את חתימת הפונקציה, כך שהיא תקבל אינדקס של i, j אינטרוואל קריאה. כדי לחסוך את זה – ראה הערות על הקוד.

הסיבוכיות של המימוש הרקורסיבי היא לפחות אקספוננציאלית. הסיבה – אנחנו מחשבים שוב ושוב את אותם הערכים. נרצה להבדיל בין תכנון דינמי לממואיזה – בממואיזה נביא פטיש, נשמור הכל מחוץ לרקורסיה ונקווה לטוב, ובתכנון דינמי נתכנן את הפתרון ונבנה את זה באמצעות ממואיזה.

כמות עצי הגזירה השונים – אספוננציאלית (הרי זה שקול ללעבור באלגו' הרגיל בלי הממואיזה). אך כמות הצמתים בעץ היא כמעט n.

נשתמש בעובדה שעשינו ממואיזה (פשוט תניחו שהקוד כתוב איפשהו). נגדיר $rr = |rules|$ ו־$n = |w|$ אז משום שאנו צריכים לבחור בזכרון את $i, j, start_var$, כאשר $i \in [n], j \in [n]$ ו־r חוסם את כמות האפשרויות של $start_var$, אז סה"כ גודל הטבלה יהי n^2r לכל היותר. זמן הריצה יהיה:

$$O\left(\sum_{i<j}\sum_{var}\sum_{i<k<j}|rules[var]|\right) = O(n^3r^2)$$

אך זה לא הדוק. נוכל לשנות את הסדר של הסיגמות (זה טריק של סכומים שכדאי להכיר):

$$O\left(\sum_{i<j}\sum_{i<k<j}\sum_{var}|rules[var]|\right) = O\left(\sum_{i<k<j}\underbrace{\sum_{var}|rules[var]|}_{O(r)}\right) = O(n^3r)$$

אומנם $rules[var] \leq r$ וגם כמות האפשרויות ל־var קטנה מ־r, ולכן זה חסום מלמעלה גם ע"י r^2. אך למעשה בשני הסכומים הללו עברנו "רק" על כל החוקים (כל var וכל אחד מהחוקים שלו)!