Predictive Parsing

$LL(1)$

Recursive Descent – חיסרון משמעותי

- הפרסר נכתב בהתאם לדקדוק : הדקדוק "צרוב" בקוד
- כלומר: לכל דקדוק יש לכתוב פרסר ייחודי
- בפרט, אם יש שינוי בדקדוק- יש לשכתב את הפרסר עצמו

דוגמה

1. S → 0 S 1
2. S → #

```
void parse_S()
{
        t = next_token();
        switch(t -> kind) {
        case 0:   { print ("Rule 1");  parse_S();
                    match(1); break }
        case #:   { print ("Rule 2");  break; }
        default:  error
        }
}
```

Predictive Parsing

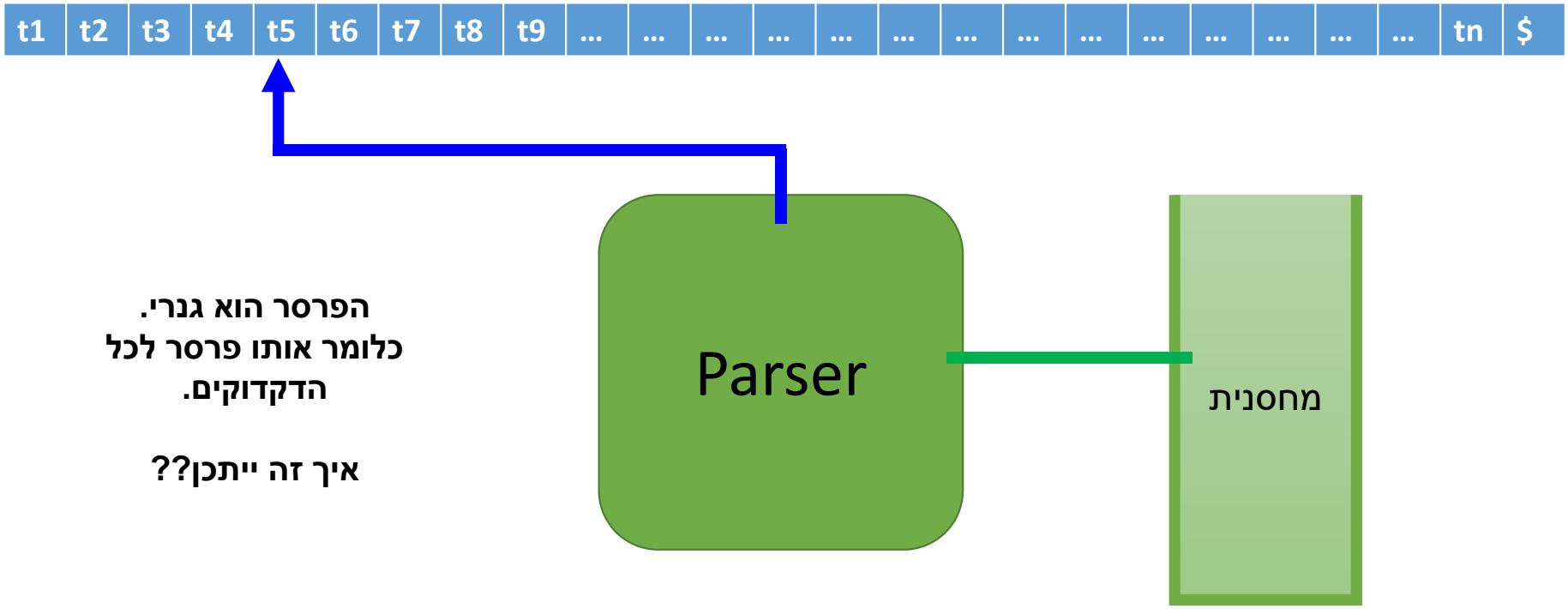
<u>תזכורת על אופן פעולתו של פרסר</u>

- בכל שלב במהלך ניתוח הקלט:
  - מעדכן תחזית להמשך (מה מצפים לראות בהמשך) – בהתאם להיסטוריה (מה היה בקלט עד עכשיו ?)
  - מוודא התאמה של המציאות (מה שמופיע בקלט) לתחזית

- **רעיון חדש - פרסר גנרי**, כלומר לא מותאם לדקדוק ספציפי.

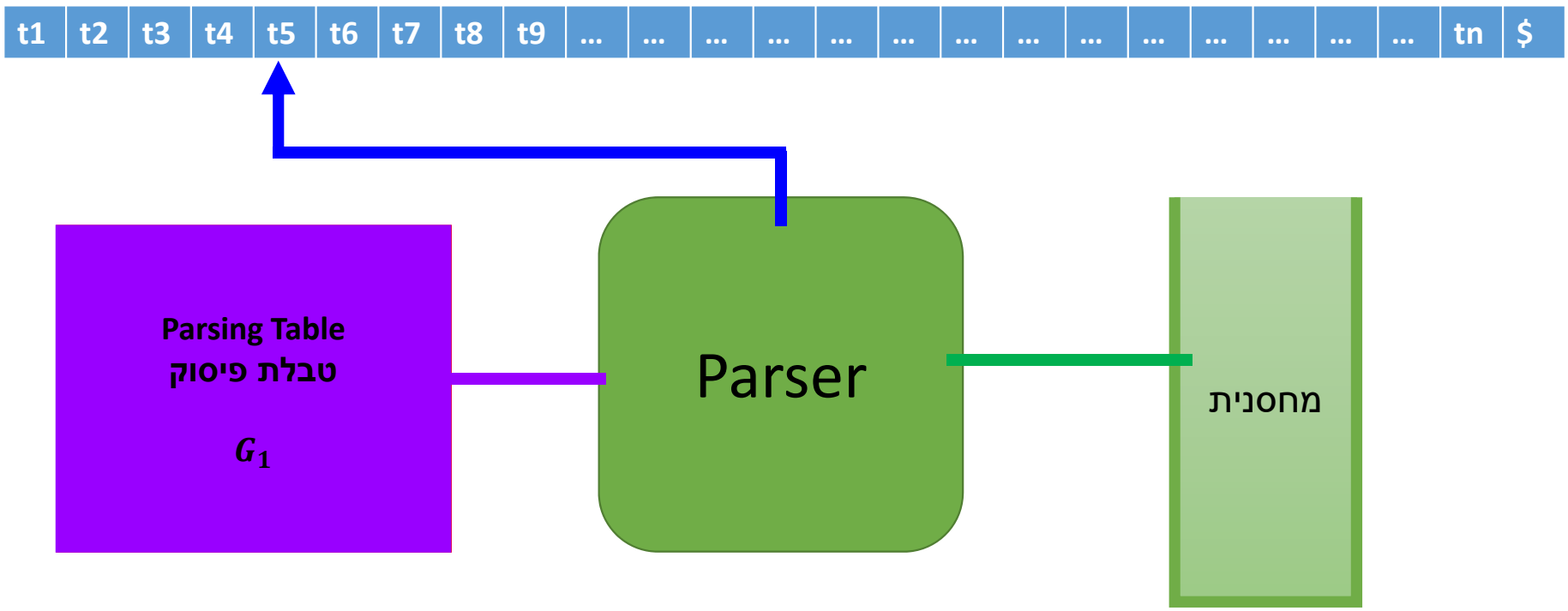
<u>אותו פרסר</u> משמש לדקדוקים שונים. מידע על הדקדוק מגיע ממקור חיצוני.

- נראה אלגוריתם בשם $LL(1)$



הפרסר הוא גנרי.
כלומר אותו פרסר לכל הדקדוקים.

איך זה ייתכן??

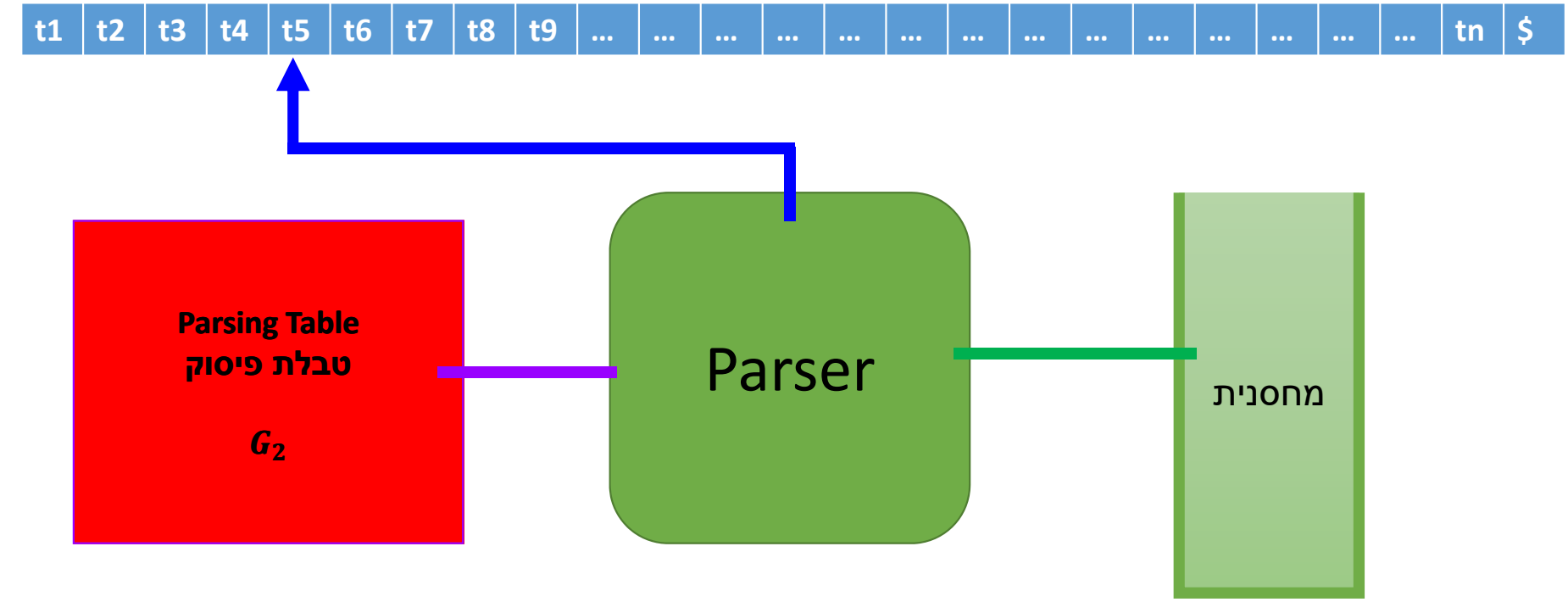


מבנה סטטי:

ריכוז מידע על חוקי תחביר

מבנה דינמי:

תחזוקת התחזית להמשך הניתוח



מבנה סטטי:

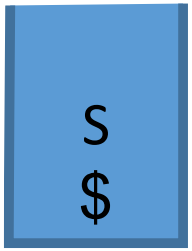
ריכוז מידע על חוקי תחביר

מבנה דינמי:

תחזוקת התחזית להמשך הניתוח

הרעיון:

- נתחזק את התחזית באמצעות מחסנית שתכיל:
  - משתנים
  - אסימונים
- תחזית התחלתית: רוצים לקרוא משהו שנגזר מ- S ואח"כ לראות $
  - בהתאם, נתחיל ממחסנית כזו:

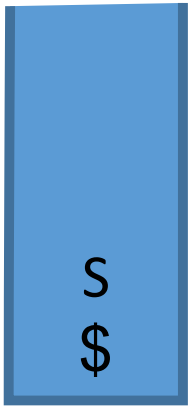


- אחרי שנסיים "לטפל" בגזירה מ-S , "נטפל" ב-$ ונרוקן את המחסנית.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$



בראש המחסנית יש S.
כלומר:
התחזית : לקרוא מהקלט משהו שנגזר מ-S.
בקלט (רק מציצים!) : 0

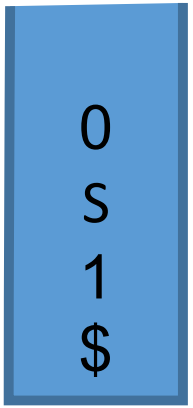
⇐ **נעדכן** את התחזית בהתאם לכלל $S \rightarrow 0S1$

דומה ל:
case 0 : Parse_S(); match(1);

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

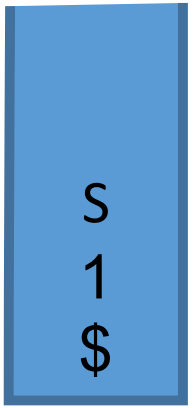


בראש המחסנית יש 0.
כלומר, התחזית : לראות בקלט 0.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : 0
⇐ נקזז את ה-0 מהמחסנית עם ה-0 מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$



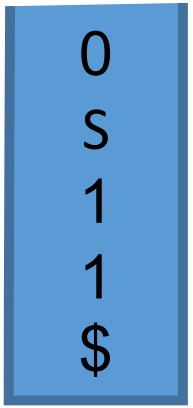
בראש המחסנית יש S.
כלומר:
התחזית : לקרוא מהקלט משהו שנגזר מ-S.
בקלט : 0

$\Leftarrow$ **נעדכן** את התחזית בהתאם לכלל $S \rightarrow 0S1$

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

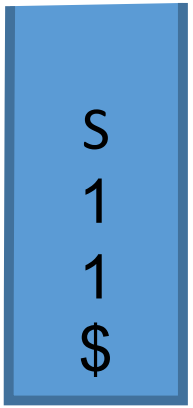


בראש המחסנית יש 0.
כלומר, התחזית : לראות בקלט 0.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : 0
⇐ נקזז את ה-0 מהמחסנית עם ה-0 מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$



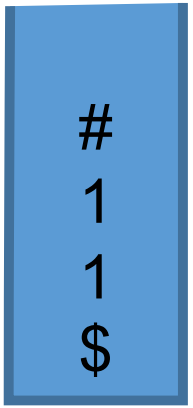
בראש המחסנית יש S.
כלומר:
התחזית : לקרוא מהקלט משהו שנגזר מ-S.
בקלט : #

$\Leftarrow$ **נעדכן** את התחזית בהתאם לכלל $S \rightarrow \#$

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

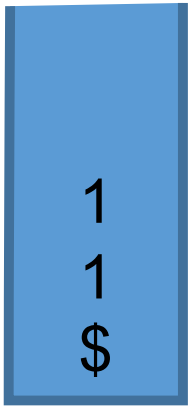


בראש המחסנית יש #.
כלומר, התחזית : לראות בקלט #.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : #
⇐ נקזז את ה-# מהמחסנית עם ה-# מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

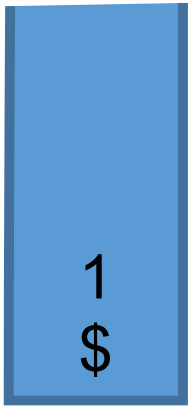


בראש המחסנית יש 1.
כלומר, התחזית : לראות בקלט 1.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : 1
⇐ נקזז את ה-1 מהמחסנית עם ה-1 מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

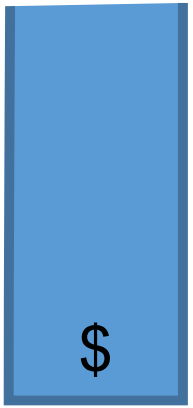


בראש המחסנית יש 1.
כלומר, התחזית : לראות בקלט 1.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : 1
⇐ נקזז את ה-1 מהמחסנית עם ה-1 מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: 00#11$

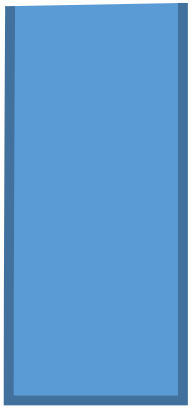


בראש המחסנית יש $.
כלומר, התחזית : לראות בקלט $.
התחזית היא **מיידית**.
נבדוק מה יש בקלט. אכן : $
⇐ נקזז את ה-$ מהמחסנית עם ה-$ מהקלט.

דוגמא

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

INPUT: $00\#11\$$



המחסנית התרוקנה:
סיימנו בהצלחה!!!

נשים לב:
המחסנית יכולה להתרוקן רק
אחרי קריאה של $

פעולה	קלט (מה שנותר)	מחסנית (תחזית)
השוואת אסימונים	$11	$ 1 1
השוואת אסימונים	$1	$ 1
השוואת אסימונים	$	$
ACCEPT		

פעולה	קלט (מה שנותר)	מחסנית (תחזית)
הצבה לפי כלל 1	$11#00	$ S
השוואת אסימונים	$11#00	$ 1 S 0
הצבה לפי כלל 1	$11#0	$ 1 S
השוואת אסימונים	$11#0	$ 1 1 S 0
הצבה לפי כלל 2	$11#	$ 1 1 S
השוואת אסימונים	$11#	$ 1 1 #

כאשר בראש המחסנית יש אסימון

- זו תחזית מיידית.
- בודקים האם המציאות תואמת את התחזית:
- משווים בין האסימון שבקלט והאסימון שבראש המחסנית.
- אם יש שוויון:
  - Pop - מוצאים את האסימון מהמחסנית
  - מוציאים את האסימון מהקלט (התקדמות בקלט ע"י $Next\_Token$)
- אחרת:
  - $Error$

כאשר בראש המחסנית יש משתנה

- בראש המחסנית מופיע משתנה A
- מציצים על האסימון הבא בקלט- t
- מסתכלים בטבלה בתא $Tab[A,t]$ ופועלים לפי מה שרשום שם.

- מה רשום שם???
- כלל גזירה מתאים $\boldsymbol{A} \rightarrow \alpha$
- ואז מבצעים:
- Pop (מוציאים את A מהמחסנית)
- $Push(\alpha)$ (מעדכנים את התחזית להיות α)

עבור הדוגמא שראינו:

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

	0	1	#	$
S	$\boldsymbol{S \rightarrow 0S1}$	$ERROR$	$\boldsymbol{S} \rightarrow \#$	$ERROR$

במקום לרשום "$ERROR$" , אפשר להשאיר תא ריק:

	0	1	#	$
S	$\boldsymbol{S \rightarrow 0S1}$		$\boldsymbol{S} \rightarrow \#$	

אלגוריתם $LL(1)$:

- איתחול:

$Push(\$)$
$Push(S)$
$t = Peek(\quad)$ // **הצצה** על האסימון הראשון בקלט

- כל עוד המחסנית לא-ריקה:

$X = Top(\quad)$ // **הצצה** על ראש המחסנית
$t = Peek(\quad)$ // **הצצה** על האסימון הנוכחי בקלט

(המשך....)

אם X הוא אסימון

- בדוק האם $t == X$
- אם כן:

$Pop(\quad)$ // הוצאת ראש המחסנית

$Next\_Token(\quad)$ // התקדמות בקלט

- אחרת:

$Error$

אם X הוא משתנה

- בדוק מה כתוב ב- $Tab[X,t]$
- אם התא ריק/כתוב בו "$error$":

$Error$

- אחרת, בהכרח כתוב בו "$\boldsymbol{X} \to \alpha$". בצע:

$Pop(\quad)$

$Push(\alpha)$

כאשר דוחפים את כל מה שמופיע ב- α **מימין לשמאל**,
כלומר מהסוף להתחלה, ואם $\alpha = \varepsilon$ לא דוחפים כלום.

אם X הוא משתנה

- בדוק מה כתוב ב- $Tab[X,t]$
- אם התא ריק/כתוב בו "$error$":

$Error$

- אחרת, בהכרח כתוב בו "$\boldsymbol{X} \to \alpha$". בצע:

$Pop(\quad)$

אם $\alpha = x_1 x_2 x_3 \cdots x_n$ בצע:

$Push(x_n)$
$Push(x_{n-1})$
$\cdots$
$Push(x_1)$

- **Init:** $\boldsymbol{Push}(\$)$; $\boldsymbol{Push}(\boldsymbol{S})$; $\boldsymbol{t} = \boldsymbol{Peek}(\quad)$;

- **While** $\boldsymbol{NotEmpty}(\boldsymbol{Stack})$:

  $\boldsymbol{X} = \boldsymbol{Top}(\quad)$;

  $\boldsymbol{t} = \boldsymbol{Peek}(\quad)$;

  if $\boldsymbol{X}$ **is a token:**

  if $\boldsymbol{X} == \boldsymbol{t}$:

  $\boldsymbol{Pop}(\quad)$;

  $\boldsymbol{Next\_Token}(\quad)$;

  else:

  Error

  if $\boldsymbol{X}$ **is a variable:**

  if $\boldsymbol{Tab}[\boldsymbol{X}, \boldsymbol{t}] = "\boldsymbol{X} \rightarrow \boldsymbol{x_1 x_2} \cdots \boldsymbol{x_{n-1} x_n}$"**;**

  $\boldsymbol{Pop}(\quad)$;

  $\boldsymbol{Push}(\boldsymbol{x_n})$;

  $\boldsymbol{Push}(\boldsymbol{x_{n-1}})$;

  $\ldots$

  $\boldsymbol{Push}(\boldsymbol{x_1})$;

  else (i.e., $\boldsymbol{Tab}[\boldsymbol{X}, \boldsymbol{t}]$ **is empty):**

  Error

- **if** $\boldsymbol{IsEmpty}(\boldsymbol{Stack})$: $\boldsymbol{ACCEPT}$

דוגמא בקובץ נפרד

מבנה של טבלת הפיסוק

- שורות: משתנים
- עמודות: אסימונים (כולל האסימון $)
- בכל תא בטבלה: כלל-גזירה.
- תא ריק: $error$
- בשורה של משתנה X יופיעו כללי גזירה מהצורה $\boldsymbol{X} \rightarrow \alpha$
- נעבור על כל הכללים, ולכל כלל נחליט באיזה תא יש לרשום אותו
- אם יש תא שמכיל יותר מאשר כלל אחד:
  הדקדוק פשוט לא מתאים לשיטה הזו ....

איך מחשבים את טבלת הפיסוק?

נתבונן בכלל $X \to \alpha$

- אם α מתחיל באסימון t: $X \to t\ \alpha_1 \dots \alpha_n$
  - נכתוב את הכלל בתא $Tab[X,t]$.

- אם α מתחיל במשתנה Y: $X \to Y\ \alpha_1 \dots \alpha_n$
  - לכל $t \in First(Y)$, נכתוב את הכלל בתא $Tab[X,t]$.

- אם α אפיס (כלומר $\alpha = \varepsilon$ או α היא סדרה של משתנים אפיסים)
  - לכל $t \in Follow(X)$, נכתוב את הכלל בתא $Tab[X,t]$.

הכללה של $Nullable$ ו- $First$

- הגדרנו את הפרדיקט $Nullable$ ואת קבוצות $First$ עבור **משתנים**.
- ניתן להכליל את שתי ההגדרות עבור **מחרוזות של משתנים ואסימונים**.

- עבור $\alpha \in (V \cup T)^*$:
- α אפיסה אם ניתן לגזור מ-α את ε
- $First(\alpha)$ היא קבוצת כל האסימונים שיכולים להופיע בתחילת משהו שנגזר מ- α

איך מחשבים את טבלת הפיסוק?

נתבונן בכלל $X \to \alpha$

- לכל $t \in First(\alpha)$: נכתוב את הכלל בתא $Tab[X,t]$.
- אם α אפיסה : לכל $t \in Follow(X)$, נכתוב את הכלל בתא $Tab[X,t]$.
- קונפליקטים
- אם מתקבל תא שמופיע בו יותר מאשר כלל גזירה אחד :

אזי הדקדוק לא מתאים לשיטת $LL(1)$ (ואומרים ש "הוא לא $LL(1)$").

דמיון לשיטת Recursive Descent

Recursive Descent
(code driven)

- פונקציה $Parse\_X$
- case t עבור כל אסימון t

- לכל $t \in First(X)$ - עדכון התחזית
עבור "$case\ t$:" : קריאות לפונקציות
parse ו- match
בהתאם לצד ימין של כלל גזירה הרלוונטי

- אם X אפיס:
לכל $t \in Follow(X)$ - עדכון התחזית
ב "$case\ t$:" - בהתאם לכלל
שיגזור מ-X את ε

$\boldsymbol{LL(1)}$
(data driven)

- שורה של X בטבלה
- עמודה של t בטבלה

- לכל $t \in First(X)$ - עדכון התחזית
דוחפים את צד ימין של כלל גזירה *Tab[X,t]* למחסנית

- אם X אפיס:
לכל $t \in Follow(X)$ - עדכון התחזית:
דוחפים למחסנית את כלל הגזירה *Tab[X,t]*
שיגזור מ-X את ε

עבור הדוגמא שראינו:

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$

```
void parse_S()
{
        t = next_token();
        switch(t -> kind) {
        case 0:    { print ("Rule 1");  parse_S();
                     match(1); break }
        case #:    { print ("Rule 2");  break; }
        default:  error
        }
}
```

	0	1	#	$
S	$\boldsymbol{S \rightarrow 0S1}$		$\boldsymbol{S} \rightarrow \#$	

עוד דוגמא :

1. $S \rightarrow 0\,S\,1$
2. $S \rightarrow \#$
3. $S \rightarrow \varepsilon$

```
void parse_S()
        t = next_token();
        switch(t -> kind) {
        case 0:     { print ("Rule 1");  parse_S();
                      match(b); break }
        case #:     { print ("Rule 2");  break; }
        case 1:
        case $:
                    { print ("Rule 3");
                      t = back_token();  break; }
        }
}
```

	0	1	#	$
S	$\boldsymbol{S \rightarrow 0S1}$	$\boldsymbol{S} \rightarrow \varepsilon$	$\boldsymbol{S} \rightarrow \#$	$\boldsymbol{S} \rightarrow \varepsilon$

דוגמא (גם ב *LL(1)* חייבים לבצע את הסילוקים המוכרים !)

$$
\begin{aligned}
S &\to ABA \mid AC \\
A &\to Aa \mid d \\
B &\to bAb \\
C &\to cC \mid \varepsilon
\end{aligned}
$$

$\xRightarrow{\text{סילוק רקורסיה שמאלית}}$

$$
\begin{aligned}
S &\to ABA \mid AC \\
A &\to dA' \\
A' &\to aA' \mid \varepsilon \\
B &\to bAb \\
C &\to cC \mid \varepsilon
\end{aligned}
$$

$\Downarrow$ סילוק גורמים שמאליים משותפים

$$
\begin{aligned}
S &\to AS' \\
S' &\to BA \mid C \\
A &\to dA' \\
A' &\to aA' \mid \varepsilon \\
B &\to bAb \\
C &\to cC \mid \varepsilon
\end{aligned}
$$

1. $S \rightarrow AS'$
2. $S' \rightarrow BA$
3. $S' \rightarrow C$
4. $A \rightarrow dA'$
5. $A' \rightarrow aA'$
6. $A' \rightarrow \varepsilon$
7. $B \rightarrow bAb$
8. $C \rightarrow cC$
9. $C \rightarrow \varepsilon$

	Nullable	First	Follow
S	-	d	$
S'	+	b, c	$
A	-	d	$b, c, \$$
A'	+	a	$b, c, \$$
B	-	b	d
C	+	c	$

	a	b	c	d	$
S				1	
S'		2	3		3
A				4	
A'	5	6	6		6
B		7			
C			8		9

קונפליקט - דוגמא

- $S \rightarrow aA \mid bAc$
- $A \rightarrow c \mid \varepsilon$

	Nullable	First	Follow
$\boldsymbol{S}$	-	a, b	$\$$
$\boldsymbol{A}$	+	c	$c, \$$

	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$	$\$$
$\boldsymbol{S}$	$S \rightarrow aA$	$S \rightarrow bAc$		
$\boldsymbol{A}$			$A \rightarrow c$ $A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$

למה " $LL(1)$ " ??

- L - קוראים את הקלט משמאל לימין
- L - הגזירה היא שמאלית-ביותר
- 1 - lookahead של אסימון אחד. כלומר, מציצים אסימון אחד קדימה

- $LL(k)$ – משפחה של אלגוריתמים כנ"ל כאשר טבלת הפיסוק מבוססת על הצצה של k אסימונים קדימה.