

# EARLEY'S PARSING ALGORITHM

אלגוריתם לניתוח תחבירי עבור דקדוקים נתון.

- אלגוריתם בתכנון דינאמי.
- שילוב של חיזוי top-down ו-bottom-up.
- אין חזרות על חישובים (תכנון דינאמי)
- מטפל היטב ברקורסיה שמאלית וחוקי אפסילון
- סיבוכיות:  $O(n^3)$

הגדרות:

**Dotted Rules**: יהיו  $\alpha\beta$  רצף של טרמינלים + נון-טרמינלים (כולל אפסילון) ויהי A נון-

טרמינל ונתון חוק בדקדוק  $A \rightarrow \alpha\beta$ , אזי  $A \rightarrow \alpha \bullet \beta$  הוא Dotted Rule.

**קשת**: אם  $A \rightarrow \alpha \bullet \beta$  הוא Dotted Rule ו- $i, j$  הם אינדקסים בטווח מחרוזת הקלט, אז

$[i, A \rightarrow \alpha \bullet \beta, j]$  יוגדר כקשת.

**קשת אקטיבית**:  $[i, A \rightarrow \alpha \bullet \beta, j]$  כך ש  $\beta \neq \epsilon$

**קשת פסיבית (או שלמה)**:  $[i, A \rightarrow \alpha \bullet \beta, j]$  כך ש  $\beta = \epsilon$

**תחילת האלגוריתם**:  $[0, S' \rightarrow \bullet S, 0]$

**סוף האלגוריתם**:  $[0, S' \rightarrow S \bullet, n]$

**פעולות האלגוריתם**: Scan, Predict, Complete

על מנת להבין את פעולות האלגוריתם, תחילה נכיר 3 פרוצדורות:

**Right\_Sister**: נתונה קשת אקטיבית  $[i, A \rightarrow \alpha \bullet B\beta, j]$ , החזר את כל Dotted Rules כך ש-

$$B \rightarrow \bullet \gamma$$

**Left\_Sister**: נתונה קשת פסיבית  $[i, A \rightarrow \gamma \bullet, j]$ , החזר את כל Dotted Rules כך ש-

$$B \rightarrow \alpha \bullet A\beta$$

**Combination**:  $[i, A \rightarrow \alpha \bullet B\beta, k] * [k, B \rightarrow \gamma \bullet, j] = [i, A \rightarrow \alpha B \bullet \beta, j]$

SCAN

$$[i, A \rightarrow \alpha \bullet w_{j+1} \beta, j] \Rightarrow [i, A \rightarrow \alpha w_{j+1} \bullet \beta, j+1]$$

PREDICT

$$[i, A \rightarrow \alpha \bullet B \beta, j] \Rightarrow [j, B \rightarrow \gamma \bullet, j]$$

תוך שימוש בחוק  $B \rightarrow \gamma$

COMPLETE

$$[i, A \rightarrow \alpha \bullet B \beta, k] \text{ AND } [k, B \rightarrow \gamma \bullet, j] \Rightarrow [i, A \rightarrow \alpha B \bullet \beta, j]$$

**האלגוריתם:**

Parse ::

```
enteredge([0, S' -> • S , 0])
```

```
for j := 1 to n do
```

```
  for every rule A -> wj do
```

```
    enteredge([j-1, A -> wj •, j])
```

```
if S' -> S • o C[0,n] then accept else reject
```

```
enteredge(i, edge, j) ::
```

```
if edge o C[i,j] then /* occurs check */
```

```
  C[i,j] := C[i,j] o {edge}
```

```
  if edge is active then /* predict */
```

```
    for edge' o rightsisters(edge) do
```

```
      enteredge([j, edge', j])
```

```
  if edge is passive then /* complete */
```

```
    for edge' o leftsisters(edge) do
```

```
      for k such that edge' o C[k,i] do
```

```
        enteredge([k, edge' * edge, j])
```

## תרגיל:

נתון הדקדוק הבא מעל ארבעת הטרמינלים {יוסי, מצא, דג, שמן}

				<--	מ
		צפ	שם	<--	צפ
		צש	פ	<--	צש
ש		תואר	ש	<--	שם
			יוסי	<--	פ
	דג		מצא	<--	תואר
			שמן	<--	ש
	שמן		דג	<--	

**המשפט: יוסי מצא דג שמן**

הרץ אלגוריתם Earley.