Parsewald Algorithmus :: Definition

- Gleiche Vorgehensweise wie CKY
- Definition von Wald (Graphentheorie): Menge von Bäumen

- Eingabe:
  - Satz $w = w_1 w_2 w_3 \ldots w_n$
  - Grammatik in Chomsky-Normalform

- Ausgabe:
  - Menge aller grammatischen Syntaxbäume
  - **Alle** Teilanalysen und bei Erfolg Syntaxbäume mit dem Topknoten S

Parsewald Algorithmus :: Code

forest [, ,] = Ø

for each s:= 1 … n **do**

  for each $A \rightarrow w_s$ **do**

    forest [s, A, s+1] := $\{A \rightarrow w_s\}$

for each l:= 2 … n **do**

  for each s:= 1 … n + 1 - l **do**

    for each t:= 1 … l - 1 **do**

      for each $A \rightarrow BC$ **do**

        forest [s, A, s+l] := forest [s, A, s+l] +

A

forest [s, B, s+t]   forest [s+ t, C, s+l]

Parsewald Algorithmus :: Erläuterung

forest [, ,] = ∅	Initialisierung der Menge mit der leeren Menge

Parsewald Algorithmus :: Erläuterung

forest [, ,] = Ø

for each s:= 1 … n **do**

for each $A \longrightarrow w_s$ **do**

forest [s, A, s+1] := $\{A \longrightarrow w_s\}$

Auflösen der Zuordnungen aus dem Lexikon

Parsewald Algorithmus :: Erläuterung

Bottom-up aufbauen der Syntaxbäume

for each l:= 2 … n **do**

for each s:= 1 … n + 1 - l **do**

for each t:= 1 … l - 1 **do**

for each A → BC **do**

forest [s, A, s+l] := forest [s, A, s+l] +

A

forest [s, B, s+t] forest [s+ t, C, s+l]

Parsewald Algorithmus :: Erläuterung

forest [, ,] = Ø

for each s:= 1 ... n **do**

for each $A \rightarrow w_s$ **do**

forest [s, A, s+1] := $\{A \rightarrow w_s\}$

Disjunkte Mengenvereinigung, d.h. hier wird der vorhandenen Menge in

forest [s, A, s+x]

ein Baum hinzugefügt.

forest [s, A, s+l] := forest [s, A, s+l] $\boxed{+}$

A

forest [s, B, s+t] forest [s+ t, C, s+l]

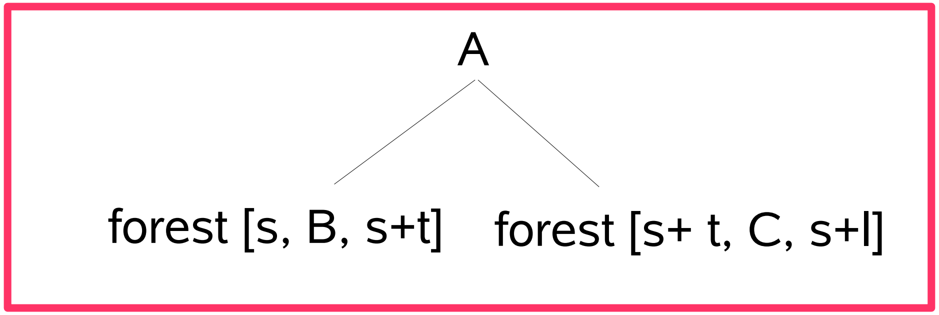
Parsewald Algorithmus :: Erläuterung

Diese Notation bedeutet, dass hier für die Regel A → BC **alle Kreuzungen** aus den in forest [s, B, s+t] und forest [s+t, C, s+l] enthaltenen Bäumen zusammen mit einem Topknoten A gebildet wird.

Kreuzprodukt (Mathematik): {A,B} x {C,D} = {AxC, AxD, BxC, BxD}

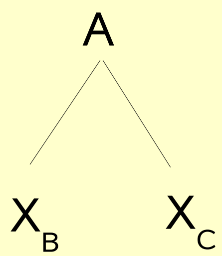
mit AxB ≠ BxA

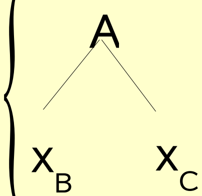
und {A,B} x {Ø} = {AxØ, BxØ} = {Ø} **!**



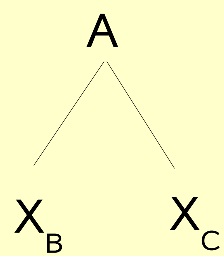
Parsewald Algorithmus :: Erläuterung

Allgemein:

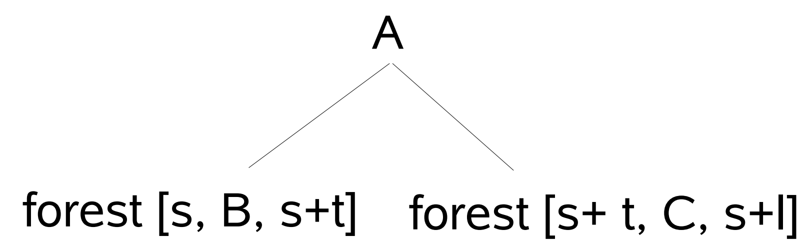


bezeichnet die Menge an Bäumen $\{$ 

$\mid x_B \in X_B \;\; x_C \in X_c \}$

Ist X_B oder X_c leer, ist natürlich auch 

leer.



Parsewald Algorithmus :: Erläuterung

Allgemein:

$$A \Rightarrow^* w_s = \begin{cases} \{ A \rightarrow w_A \} & \text{für } |w_A| = 1 \\[1em] \displaystyle\sum_{\substack{A \rightarrow BC \\ w_A = w_B w_C}} \begin{array}{c} A \\ B \Rightarrow^* w_B \quad C \Rightarrow^* w_C \end{array} & \text{für } |w_A| > 1 \end{cases}$$

$$\begin{array}{c} A \\ \text{forest } [s, B, s+t] \quad \text{forest } [s+t, C, s+l] \end{array}$$

Parsewald-Algorithmus :: Beispiel

for each s:= 1 … n **do**

for each $A \longrightarrow w_s$ **do**

forest [s, A, s+1] := $\{A \longrightarrow w_s\}$

```
forest:

[1,N,2]      => N -> they
[2,trans,3]  => trans -> study
[3,trans,4]  =>  trans -> fish
[3,N,4]      =>  N ->fish
[4,prep,5]   =>  prep -> in
[5,N,6]      =>  N -> cans
```

Parsewald Algorithmus :: Beispiel

```
nach l=2 wird forest erweitert um:

[2,intr,4] =     intr              [4,intr,6] =     PP
               /      \                           /    \
              N       trans                     prep    N
              |         |                        |      |
            study     fish                       in     cans
```

for each l:= 2 … n **do**

for each s:= 1 … n + 1 - l **do**

for each t:= 1 … l - 1 **do**

for each A → BC **do**

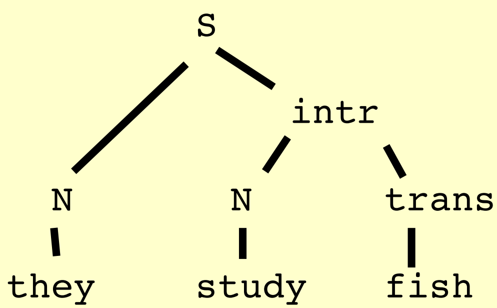
forest [s, A, s+l] := forest [s, A, s+l] +

```
                    A
                 /     \
forest [s, B, s+t]     forest [s+ t, C, s+l]
```

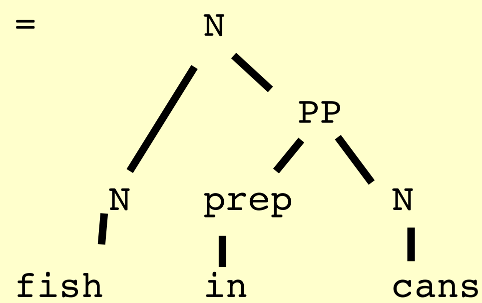
Parsewald Algorithmus :: Beispiel

nach l=3 wird forest erweitert um:

[1,intr,4] =



[3,intr,6] =



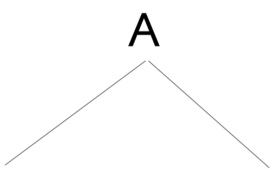
for each l:= 2 ... n **do**

for each s:= 1 ... n + 1 - l **do**

for each t:= 1 ... l - 1 **do**

for each A → BC **do**

forest [s, A, s+l] := forest [s, A, s+l] +

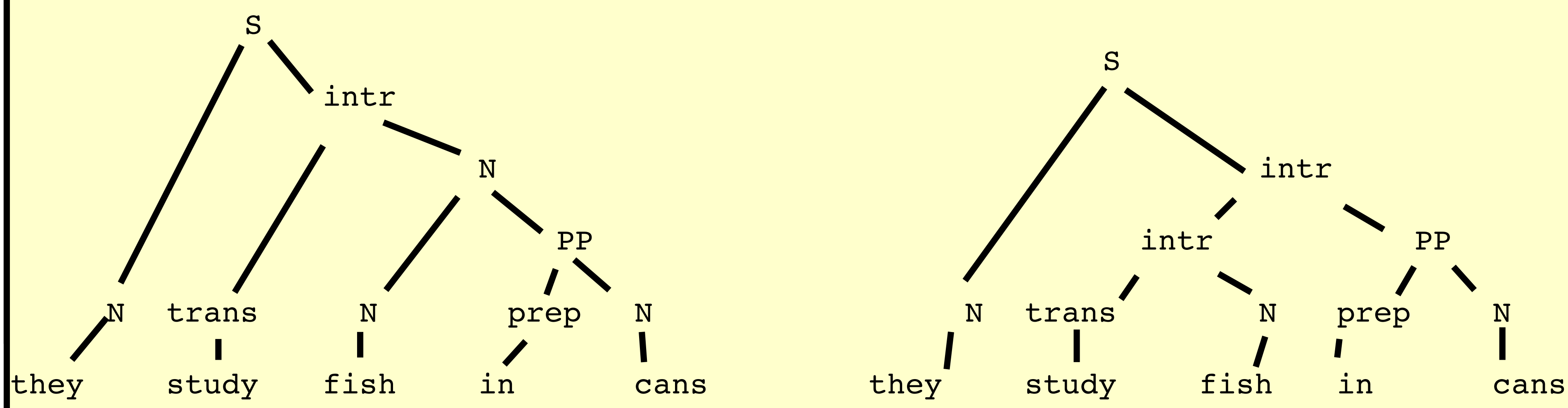


forest [s, B, s+t] forest [s+ t, C, s+l]

Parsewald Algorithmus :: Beispiel

```
nach l=5 wird forest erweitert um:

[1,S,6] =
```



Natürlich „wandert" der Algorithmus genauso die Chart-Tabelle nach oben, wie der CKY.
Nur das sich in den Feldern, dann die Bäume befinden.

Parsewald Algorithmus :: Zusammenfassung

- Parsewald
  - verwaltet Teilanalysen
    - z.B. nützlich um heuristische Analysen mit abgespeicherten Zwischenergebnissen zu machen
    - hat eine einfache aber mächtige Definition zum Erzeugen von Syntaxbäumen (Kreuzprodukt)
  - hat die gleiche Komplexität wie CKY $O(n^3)$