Formale Sprachen: EBNF

-> Grammatik:

$G = (N; T; P; S)$ mit

$N = \{N1; N2; N3\}$

$T = \{"A"; "B"\}$

$P = \{$

N1 = "A" | "B".

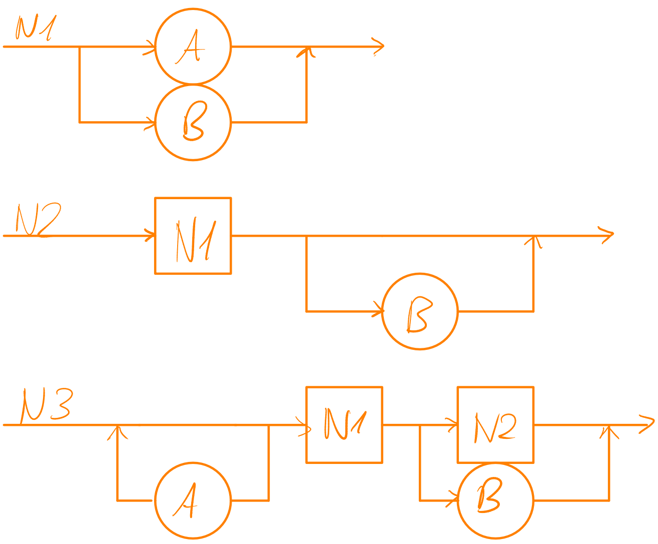
N2 = N1 ["B"].

N3 = {"A"} N1 (N2 | "B").

$\}$

$S = N3$

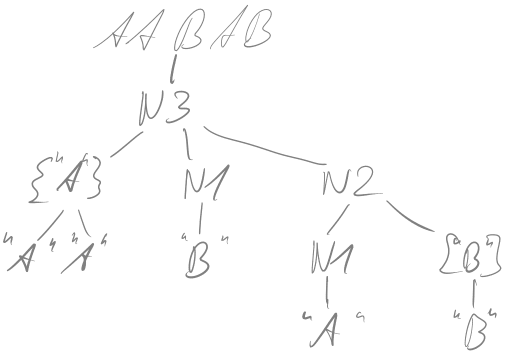
-> Syntax Diagramm



($\varepsilon \hat{=}$ leeres Wort ; $|N3| \hat{=}$ Länge von N3)

Ausdrücke Ableiten:

Baum-Schreibweise:



Ableitungs-Schreibweise:

N3 = {"A"} N1 (N2 | "B") =

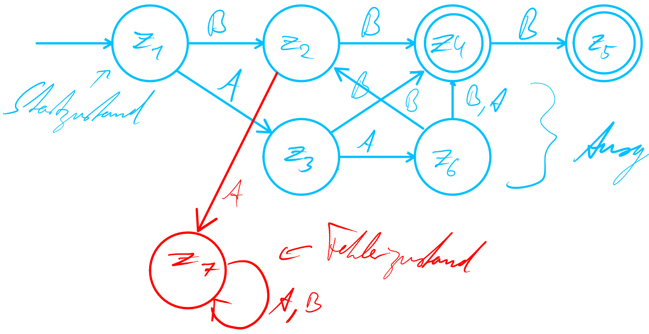
= "A" "A" ("A" | "B") N2 =

= "A" "A" "B" N1 ["B"] =

= "A" "A" "B" ("A" | "B") "B" =

= "A" "A" "B" "A" "B"

-> Endliche, erkennende Automaten:



<- Endzustand -> Wort akzeptiert

Ausgang aus einem Zustand nicht eindeutig

-> nicht deterministischer Automat

Java Codebeispiel für einen erkennenden Automaten:

```
class Automat {
    int zustand;
    public Automat ( ) { };
    public boolean zeichenkettePruefen ( char [ ] wort ) {
        zustand = 1;
        for (int i = 0; i < wort.length; i ++){
            uebergang (wort [i]);
        }
        if ( zustand == 4 || zustand == 5) return true;
        return false;
    }
    private void uebergang (char c) {
        switch (zustand) {
            ...
            case 1:
                switch ( c) {
                    case 'A':
                        zustand = 3;
                        break;
                    case 'B':
                        zustand = 2;
                        break;
                }
                break;
            ...
        }
    }
}
```