2. Beispiel: $L(G) = L(a^* b^* c^+)$

$S \to ABC$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

$C \to cC \mid c$

$S \to ABC$:

$First(ABC) = First(A) \setminus \{\varepsilon\} \cup First(B) \setminus \{\varepsilon\} \cup First(C)$

$= \{a, b, c\}$

$A \to aA$:

$First(aA) = \{a\}$

$A \to \varepsilon$:

$Follow(A) = First(BC) = \{b, c\}$

$B \to bB$:

$First(bB) = \{b\}$

$B \to \varepsilon$

$Follow(B) = First(C) = \{c\}$

	a	b	c	$
S	S->ABC	S->ABC	S->ABC	error
A	A->aA	A->epsilon	A->epsilon	error
B	error	B->bB	B->epsilon	error
C	error	error	C->cC, C->c	error

Syntaxanalyse

Konstruktion der Parsertabelle

- Die Parsertabelle TAB enthält für jedes Nonterminalsymbol A eine Zeile und für jedes Terminalsymbol x eine Spalte.
- Ein Eintrag TAB(A,x) enthält die Regel(n), die für A angewendet werden, falls x das Vorausschau-Token ist.

Algorithmus zur Konstruktion der Parsertabelle

Für jede Ableitungsregel A -> α :

1. Für jedes Terminalsymbol a $\in$ FIRST(α) übernehme die Regel A -> α in TAB[A, a]
2. Falls $\varepsilon \in$ FIRST(α), übernehme Regel A -> α in TAB[A, b] für jedes b $\in$ FOLLOW(A) Falls $\varepsilon \in$ FIRST(α) und \$ $\in$ FOLLOW(A), übernehme Regel A -> α in TAB[A, \$].

$C \to cC$

$First(cC) = \{c\}$ ┐

$C \to c$ ├ LL(1) - Eigenschaft

$First(c) = \{c\}$ ┘ nicht erfüllt!

Wie könnte man einen Top-down-Parser schreiben

für diese Sprache:

- 2 Vorausschaumsymbole lesen (geht hier, aber nicht immer)
- stattdessen eine andere Grammatik verwenden, die dieselbe Sprache erzeugt

neue Grammatik:

$S \to ABC$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$
$C \to cD$
$D \to cD \mid \varepsilon$

Parsetabelle:

	a	b	c	$
S	S->ABC	S->ABC	S->ABC	error
A	A->aA	A->epsilon	A->epsilon	error
B	error	B->bB	B->epsilon	error
C	error	error	C->cD	error
D	error	error	D->cD	D->epsilon

$C \to cD$
$First(cD) = \{c\}$
$D \to cD$
$First(cD) = \{c\}$
$D \to \varepsilon$
$Follow(D) = Follow(C) = Follow(S) = \{\$\}$

LL(1) - Eigenschaft erfüllt!
eindeutige Ableitungsbaum!

Ableitungsbaum für $w = abcc\$$

