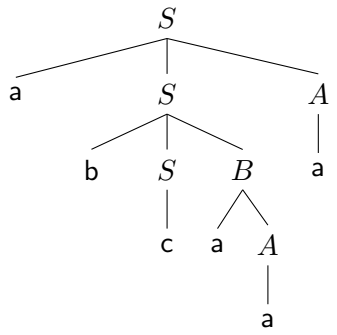
1) *Lösung.* –



- G_1 ist eindeutig, da es für jedes Wort $x \in \mathsf{L}(G_1)$ nur genau eine Linksableitung gibt. Mit den Regeln für S erhält man jeweils nur Wörter mit unterschiedlichen Präfixen, alleine mit diesen Regeln könnte also keine Mehrdeutig entstehen. Der hintere Teil der Wörter (der aus den As und Bs abgeleitet wird) kann auch nicht mit unterschiedlichen Linksableitungen erzeugt werden.
- $G_2 = (\{S\}, \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, R, S)$, wobei die Regeln R wie folgt definiert sind:

$$S \to \mathsf{a}\mathsf{a}S\mathsf{b} \mid \mathsf{c}$$

□

2) a) Die folgende Tabelle zeigt die rekursive Inferenz für den String *abab*:

Schritt	Variable	Regel	Rekursion
1	b	$\mathsf{B} \to b$	
2	a	$\mathsf{S} \to a$	
3	ab	$\mathsf{S} \to \mathsf{SB}$	1,2
4	bab	$\mathsf{B} \to \mathsf{BS}$	1, 3
5	abab	$\mathsf{S} \to \mathsf{SB}$	2, 4

b) Zum Beispiel gibt es für das Wort b zwei unterschiedliche Linksableitungen:

$$S \Rightarrow \mathsf{SB} \Rightarrow \mathsf{B} \Rightarrow b$$
$$S \Rightarrow \mathsf{SB} \Rightarrow \mathsf{B} \Rightarrow \mathsf{BS} \Rightarrow b\mathsf{S} \Rightarrow b$$

3) `ex.l`

```
%{
  #include "y.tab.h"
%}
```

```
%%
[a-z]   return ATOM;
"("     return LB;
")"     return RB;
"T"     return TRUE;
"F"     return FALSE;
"not"   return NEG;
"and"   return AND;
"or"    return OR;
"imp"   return IMP;
[ \t] ; /* skip whitespace */
\n      return ENTER;
%%
```

ex.y

```
%{
#include <stdio.h>
void yyerror(const char *);
int yylex();
int yyparse();
%}

%token ATOM TRUE FALSE LB RB NEG AND OR IMP ENTER

%%

prog :  prop ENTER { printf("Accepted\n"); return 0; }
        ;

prop:   atom
        | NEG prop
        | LB prop AND prop RB
        | LB prop OR prop RB
        | LB prop IMP prop RB
        ;

atom:   ATOM
        | TRUE
        | FALSE
        ;

%%

void yyerror(const char *str) {
        printf("Syntax␣error\n");
}

int yywrap() {
```

```
        return 1;
}

int main() {
        printf("Enter␣expression:␣");
        yyparse();
}
```