ANHANGB: DAS IMPLEMENTIERTE SYSTEM

B.1 Programmbeschreibung

B.1.1 Programmarchitektur

Das System besteht aus drei Modulen: Morphotaktik mit Benutzerschnittstelle, Allomorphiekomponente und Morphophonologie. Die folgenden Graphiken verdeutlichen anhand von Beispielen den typischen Programmablauf. Die Ausgabe der Morphotaktik "füttert" dabei die Allomorphie, und diese wiederum die Morphophonologie.

(1)

Eingabe			**Ausgabe**
pyes prs 1sg	=>	**Morphotaktik**	=>	(((S51+K21)prs)1sg)
vras prs nak 3pl	=>		=>	((((S66+V25+K27)nak)prs)1sg)
pi adm 2pl	=>		=>	(((S75)(par))((S97+V3d+K3e)2pl)

(2)

Eingabe			**Ausgabe**
((S9+K11)1sg)	=>	**Allomorphie**	=>	laj
((S96+V3c+K3d)1sg)	=>		=>	jam
(((S93+V2o+K3b)aor)1sg)	=>		=>	pashë

(3)

Eingabe			**Ausgabe**
hap(h)em	=>		=>	hapem
pi(h)em	=>		=>	pihem
hap(ë)m(ë)	=>	**Morphophonologie**	=>	hapëm
pi(ë)m(ë)	=>		=>	pimë
mëso(i|u)	=>		=>	mësoi
vra(i|u)	=>		=>	vrau

B.1.2 Benutzung

Nach dem Aufruf von "syn|mo|phon <ENTER>"[1][2] wartet das Programm auf die Eingabe einer Feature-Spezifikation. Mögliche Features sind die Zitierformen der Verben und die in **§2/1** besprochenen morphologischen Features[3]. Die Reihenfolge der Eingabe-Features ist irrelevant, solange die Eingabe in sich konsistent ist (z.B. nicht **ind** und **opt** enthält). Für durch den Benutzer unspezifizierte Features besitzt das Programm festgelegte Default-Werte: **prs** für Tempus, **ind** für Modus und **akt** für die Diathese. Wird kein Verb angegeben, gibt das System die jeweiligen Formen von *punoj* aus. Weglassung des **per/num**-Werts führt zur Ausgabe aller 6 Formen. Widersprüchliche Eingaben korrigiert das System intern nach der Reihenfolge der Eingabe. So führt die Folge "opt imv" zur Ausgabe von **imv**-Formen, die umgekehrte Abfolge zur Ausgabe von **opt**-Formen. Dies erlaubt dem Benutzer auch während der Eingabe Korrekturen vorzunehmen. Der Abschluß der Eingabe erfolgt durch <ENTER>.

---

[1] Das selbe Ergebnis erhält man durch den Aufruf "molex".

[2] Jeder Teil-Befehl entspricht einer der oben genannten System-Komponenten. So lassen sich mit "syn" morphotaktische Strukturen generieren.

[3] s. auch das Hauptprogramm in "syn.l".

Um größere Ausschnitte des Paradigmas zu erhalten, sind einige vordefinierte Shell-Skripts[4] vorhanden. "aza <ZITIERFORM> <ENTER> "generiert die wichtigsten Formen für <ZITIERFORM> die Zitierform eines Verbs. "kli1 <ZITIERFORM> <ENTER>" generiert repräsentative Formen mit präverbalen pronominalen Klitika. "kli2 <ZITIERFORM> <ENTER>"generiert repräsentative Formen mit postverbalen pronominalen Klitika. Die Skripts schreiben die entsprechenden Formen mit erklärendem Glossar in die Datei a.tmp, die automatisch durch den Editor vi geöffnet wird.
Der entwickelte Compiler für **reguläres (f)lex** (s. u. "phoen.l") ermöglicht es auch, eigene **mo_lex**-Theorien zu implementieren. Der Aufruf erfolgt durch "phoen <DATEI> <ENTER> für <DATEI> " eine Text-Datei, die eine Theorie in **regulärem (f)lex** enthält.

B.1.3 Implementation

Die Programme wurden auf unter AIX, Version 4.2, entwickelt. Bis auf mo.m sind alle Programme **flex**-Programme. Verwendet wurde die **flex**-Version 2.5.2.

B.1.4 Einzelne Programme

phoen.l ist der Quelltext für einen Compiler, der Programme in **regulärem (f)lex** in **(f)lex**-Programme übersetzt. syn.l entspricht der oben beschriebenen Morphotaktik, phon.l der Morphophonologie und mo.m der Allomorphie-Komponente. mo.m ist ausschließlich in **regulärem (f)lex**, phon.l durchgehend in **(f)lex**. syn.l enthält umfangreiche C-Routinen. delfi.h ist die Header-Datei für syn.l.

B.2 Quelltexte[5]

B.2.1. delfi.h

```
enum fitsch { deg, akt, nak, prs, imf, aor, prf, plq, ao2, fut, ind, kon, opt, imv, inf, par, nor, jus, ger,
        priv, abs, part, prt, sg1, sg2, sg3, pl1, pl2, pl3 };

        /* STAMM */
char stem[25] = "S1+V10+K10";       /* speichert Stamm, Default: punoj */
        /* FEATURES */
int hord = nor;                 /* H_ord: jus, fut, abs, inf */
int konn = ind;                 /* Konnektoren: kon priv ger */
char fin = 1;                   /* Finitheit: ja/nein */
char adm = 0;                   /* adm: ja/nein */
char asp = nor;                 /* Perfektivitiaet */
char dia = 1;                   /* Diathesen: akt, nak */
int lab = prs;                  /* Label: Postverbale Tempus- + Modus- Affixe  */
int saf = prs;                  /* Stammaffixe */
int per = nor;                  /* Person-Numerus: 1sg 2sg ... */
        /* KLITIKA */
int kl1 = 0;                    /* pronom. Akkusativ-Klitika, 1 = acc, 1.sg,... */
int kl2 = 0;                    /* pronom. Dativ -Klitika, 1 = dat, 1.sg,... */
int kl3 = 0;                    /* Nichtaktiv-Klitikon ja/nein */
int kl4 = 0;                    /* ethischer Dativ (Klitikon )          */
int kli_stat = 0;               /* Flag: keine Klitika, prae-, oder postverbale */
```

[4] Quelltexte für die Shell-Skripts fehlen hier aus Platzgründen.

[5] "ë" wird im Programm-text durch "E", "ç" durch "C" ersetzt.

B.2.2 syn.l

```
%{

# include <stdio.h>
# include <string.h>
# include "delfi.h"

%}

%%

        /* ************************ VERBEN ************************ */

punoj           src("S1+V10+K10");
shkruaj         src("S2+V11+K11");
ruaj            src("S3+V12+K12");
luaj            src("S4+V13+K12");
kthej           src("S5+V14+K10");
lyej            src("S6+V15+K11");
zhyej           src("S7+V16+K11");
zjej            src("S8+K10");
laj             src("S9+K11");
fshij           src("S10+K11");
brej            src("S11+K12");
rroj            src("S12+K13");
mbroj           src("S13+K13");
yej             src("S14+V1a+K12");
mbaj            src("S15+K12");
bEzaj           src("S16+K12");
buj             src("S17+K14");
arrij           src("S18+K15");
shtyj           src("S19+K1b");
bEj             src("S20+K16");
blej            src("S21+V30+K17");
ndiej           src("S22+V31+K11");
mErdhij  src("S23+K10");
dErsij          src("S24+K10");
hyj             src("S25+K18");
prij            src("S26+K19");
hap             src("S27");
dElir           src("S28");
prish           src("S29");
rrok            src("S30");
hipi            src("S31");
heq             src("S32+V21+q");
sjell           src("S33+V2q+ll");
vjel            src("S34+V21+l");
vjedh           src("S35+V21+dh");
dridh           src("S36+V22+dh");
prier           src("S37+V23+r");
fsheh           src("S38+V24+h");
ndez            src("S39+V24+z");
ngjesh          src("S40+V24+sh");
rrah            src("S41+V25+h");
njoh            src("S42+V26+h");
```

```
nxjerr          src("S43+V21+RR");
pjek            src("S44+V21+K");
djeg            src("S45+V21+G");
dal             src("S46+V2d+l");
marr            src("S47+V27+RR");
stErvit         src("S48+K20");
vendos          src("S49");
shetit          src("S50+K21");
pyes            src("S51+K23");
spErkas         src("S52+V29+K22");
trokas          src("S53+V2a+K22");
humbas          src("S54+V2b+K24");
vErres          src("S55+V2c+K24");
thErras         src("S56+V2b+K24");
shtErras src("S57+V2b+K24");
kEllas          src("S58+V2b+K24");
kElthas         src("S59+V2b+K24");
bErtas          src("S60+V2b+K24");
pErkas          src("S61+V2b+K24");
kErcas          src("S62+V2b+K24");
pElcas          src("S63+V2b+K24");
flas            src("S64+V2f+K26");
shklas          src("S65+V2b+K24");
vras            src("S66+V25+K27");
ngas            src("S67+V2g+K26");
gErgas          src("S68+V2h+K28");
shkas           src("S69+V25+K27");
pres            src("S70+V2i+K26");
dhjes           src("S71+V2j+K27");
pres2           src("S72+V2k+K28");
vdes            src("S73+V2l+K29");
loz             src("S74+K34");
pi              src("S75");
di              src("S76+K2a");
ngre            src("S77+V2m+K2a");
fle             src("S78+V2z+K2a");
zE              src("S79+V2n+K35");
vE              src("S80+V2n+K35");
shpie           src("S81+V33+K35");
shtie           src("S82+V34+K36");
lE              src("S83+V35+K37");
dua             src("S84+V36+K1a");
them            src("S85+V37+K38");
vete            src("S86+V38+K1c");
gjej            src("S87+V39+K1d");
lind            src("S88");
bie             src("S89+V3a+K36");
bie2            src("S90+V3b+K37");
ha              src("S91");
rri             src("S92+K3a");
shoh            src("S93+V2o+K3b");
vij             src("S94+V3k+K1e");
jap             src("S95+V2p+K3c");
jam             src("S96+V3c+K3d");
kam             src("S97+V3d+K3e");

        /* ******************** FEATURES *************************** */
```

```
\n              {ad_just(); tus(); exit(1);}
prs             {lab = prs; }
imf             {lab = imf; }
aor             {lab = aor; }
prf             {lab = prs; asp = prt;}
plq             {lab = imf; asp = prt;}
ao2             {lab = aor; asp = prt;}
op2             {lab = opt; asp = prt;}
fut             {hord = fut; }
jus             {hord = jus; }
ind             {hord = ind; }
kon             {konn = kon;}
opt             {lab = opt; }
imv             {lab = imv; }
akt             {dia = akt;}
nak             {dia = nak;}
priv            {konn = priv;}
ger             {konn = ger; }
inf             {hord = inf; }
abs             {hord = abs; }
prt             {asp = prt;}
1sg             per = sg1;
2sg             per = sg2;
3sg             per = sg3;
1pl             per = pl1;
2pl             per = pl2;
3pl             per = pl3;
adm             {adm = 1; }
a1s             kl2 = sg1;
a2s             kl2 = sg2;
a3s             kl2 = sg3;
a1p             kl2 = pl1;
a2p             kl2 = pl2;
a3p             kl2 = pl3;
d1s             kl1 = sg1;
d2s             kl1 = sg2;
d3s             kl1 = sg3;
d1p             kl1 = pl1;
d2p             kl1 = pl2;
d3p             kl1 = pl3;
e1s             kl4 = sg1;
e1p             kl4 = pl1;

        /* ****************** DEFAULTS ******************** */

[a-zA-Z0-9]+    {fprintf(stderr,"Ungueltige Eingabe:%s!\n",yytext); exit(1);}
[\t ]
.               ;
```

```
%%

        /* ***************** src ***************************** */

src(char *s)                            /* kopiert Staemme in Variable "stem" */
{
strcpy(stem,s);
}

        /* ***************** tus ***************************** */

tus()                                   /* steuert Ausgabemodus */
{
printf("%c",lab);
if(fin == 0 || per != nor)              /* nur eine Form */
        eksek();
else    if(lab == imv)                  /* Imperativ: 2 Formen */
                {
                 per = sg2; eksek();
                 per = pl2; eksek();
                }
else                                    /* Vollstaendiges Per/Num-Paradigma */

        for( per = sg1; per <= pl3; ++per )
                eksek();
}

        /* ***************** ad_just ***************************** */

ad_just()       /* ergaenzt und korrigiert Eingabe-Features */
{
switch(hord)                /* Diese Partikel verlangen Konj      */
        {
        case fut :
        case jus : konn = kon; fin = 1; break;
        case inf : konn = kon; fin = 0; break;
        case abs : konn = kon; fin = 0; asp = nor; break;
        default :       fin = 1;
        }

switch(konn)                /* Konjunktiv, Privativ, Gerundiv */
{
case kon:       if(fin == 1 && label())  lab = prs; break;
case priv:      fin = 0; break;
case ger: fin = 0; break;
}


if(fin == 0)                /* keine finiten Admirativ-formen */
        {
        adm = 0;
        lab = par;
        }
if(adm == 1  && label())    /* kein Adm mit (opt|imv|aor)          */
        lab = prs;

if(lab == imv && per != pl2 && per != nor)    /* mit Imv nur 2(sg|pl) */
```

```
        per = sg2;


if(dia == nak &&   asp != prt  && (label() || lab == par ||adm == 1))
        kl3 = 1;                                        /* Nichtaktivklitikon ? */
if(kl1 == 0 &&  kl2 == 0 && kl3 == 0 && kl4 == 0)
kli_stat = 0;                                           /* keine Klitika */
else if(lab != imv)
        kli_stat = 1;                                   /* Praeverbale... */
else if(kl1 == pl1 && (kl2 != 0 || kl3 != 0))
        kli_stat = 1;                                   /* ..,praeverbale... */
else
        kli_stat = 2;                                   /* ..,postverbale  */
                                                        /*    Klitika */
}
        /* ***************** eksek ***************************** */

eksek()                 /* drucke Verbform */
{
hord_print();                                           /* H_ord */
konn_print();                                           /* Konnektoren */
saf = lab;                                              /* Stammaffix = Label */
if(kli_stat == 1)
        {
        kliprint();                                     /* Praeverbale Klitika */
        }
if(asp == prt)                                          /* Perfekt */
        {
        if(adm == 1)                                    /* Admirativ Perfekt */
                {
                printf("(");
                saf = par;
                auxprint();
                saf = lab;
                }

        auxprint();                                     /* Perfekt-Auxiliare */
        if(adm == 1)
                printf(")");
        printf(" ");
        saf = par;                                      /* Prf --> Partizip */
        }
else                                                    /* Klammern */
        {
        if(adm == 1)
                printf("(");
        else
                {
                if (saf != par && saf != prs)
                printf("(");
                if((saf == prs || saf == imf) && dia == nak)
                printf("(");
                }
        }
```

```
if(adm == 1)                                    /* Nicht-Perfekt-Admirativ */
        saf = par;
printf("((%s)",stem);                           /* Stamm */

if(dia == nak && (saf == imf || saf == prs))    /* nak-Suffix */
        printf("nak)");
mainprint();                                    /* Temp/Mod-Suffixe */
if(kli_stat == 2)                               /* Postverbale Klitika */
        kliprint();
if(saf != par)
        endprint();                             /* Endungen */
if(adm == 1 && asp != prt)                      /* Admirativ-KAM */
        {
        saf = lab;
        auxprint();
        printf(")");
        }
printf("\n");
}

        /* ****************** auxprint************************ */

auxprint()              /* drucke Auxiliar */
{
if(saf == imf || saf == opt || saf == aor)      /* Klammern */
        printf("(");
if(dia == nak && adm != 1)
        printf("((S96+V3c+K3d)");
else
        {
        printf("((S97+V3d+K3e)");
        }
mainprint();
if(saf != par)
        endprint();
}

        /* ******************** mainprint ********************** */

mainprint()     /* drucke saf: prs, imf, aor, ...., par */
{
switch(saf)
        {
        case prs: break;
        case imf: printf("imf)"); break;
        case aor: printf("aor)"); break;
        case opt: printf("opt)"); break;
        case imv: printf("imv)"); break;
        case par: printf("par)"); break;
        }
}
```

```
        /* ************** endprint ****************************** */

endprint()                              /* drucke Endungen */
{
switch(per)
        {
        case sg1:       printf("1sg");break;
        case sg2:       printf("2sg");break;
        case sg3:       printf("3sg");break;
        case pl1:       printf("1pl");break;
        case pl2:       printf("2pl");break;
        case pl3:       printf("3pl");break;
        }
printf(")");
}

        /* ************** kliprint ************************** */

kliprint()                              /* drucke pronominale Klitika */
{
int k_count;
int i;
k_count = 3;
if(e_print())                           /* Ethischer Dativ */
        --k_count;
if(d_print())                           /* Dativ-Klitika */
        --k_count;
if(a_print())                           /* Akkusativ-Klitika */
        --k_count;
if(kl3)                                 /* nak-Klitikon */
        {
        printf("(nak");
        ++k_count;
        }
printf("(kli)");                        /* kli */
for( i = 0; i< k_count; ++i)  /* Klammern */
        printf(")");
if(kli_stat == 1)
        printf(" ");}

/* ***************** label ******************** */

label()                                 /* testet, ob "Modalform" vorliegt */
{
switch(lab)
        {
        case    aor :
        case    imv :
        case    opt :   return(1); break;
        default:  return(0);
        }
}
```

```
        /* ************** d_print***************************** */

d_print()       /* drucke Dativ-Klitika */
{
switch(kl1)
        {
        case    sg1: printf("(d1s"); return(0); break;
        case    sg2: printf("(d2s"); return(0); break;
        case    sg3: printf("(d3s"); return(0); break;
        case    pl1: printf("(d1p"); return(0); break;
        case    pl2: printf("(d2p"); return(0); break;
        case    pl3: printf("(d3p"); return(0); break;
        default: return(1);
        }
}

        /* ************** a_print ********************** */

a_print()       /* drucke Akkusativ-Klitika */
{
switch(kl2)
        {
        case    sg1: printf("(a1s"); return(0); break;
        case    sg2: printf("(a2s"); return(0); break;
        case    sg3: printf("(a3s"); return(0); break;
        case    pl1: printf("(a1p"); return(0); break;
        case    pl2: printf("(a2p"); return(0); break;
        case    pl3: printf("(a3p"); return(0); break;
        default: return(1);
        }
}

e_print()       /* drucke "ethischen Dativ" */
{
switch(kl4)
        {
        case    sg1: printf("(d1s"); return(0); break;
        case    pl1: printf("(d1p"); return(0); break;
        default: return(1);
        }
}

        /* ***************** hordprint ********************** */

hord_print()
{
switch(hord)                                    /* drucke H_order */
        {
        case fut : printf("(fut) "); break;
        case jus : printf("(jus) "); break;
        case inf : printf("(inf) "); break;
        case abs : printf("(abs) "); break;
        }
}
```

```
        /* ***************** konn-print ******************** */

konn_print()
{
switch(konn)                                        /* drucke Konnektoren */
        {
        case kon:           printf("(kon) "); break;
        case priv:          printf("(priv) "); break;
        case ger: printf("(ger) ");  break;
        }
}
```

B.2.3 phon.l

```
VOKAL           [aeiouyE]
JVOKAL          [aeioujyE]
DVOKAL          [aeiouyE][aeiouyE]
IK              [ehiak]
LAUT            [a-zA-Z]

%start KLI
%%

        /*  opt: paC(f)sha -->  paCa, paC(f)tE --> pastE */


C[(]f[)]sh                  printf("C");
C[(]f[)]                    printf("s");

        /* opt: ndenj(f)sh --> ndenjC , prish(f)sh  --> prishC */
        /* opt: dirs(f)sh  --> dirC , zEn(f)sh --> zEnC */

nj[(]f[)]sh                 printf("njC");
[ns][(]f[)]s h              printf("%cC",yytext[0]);
sh[(]f[)]sh                 printf("shC");


        /* par: prier(u)r(E) --> prier, dElir(u)r(E) --> dElirE        */
        /* par: nxjeRR(u)r(E) --> nxjerrE                  */

{DVOKAL}[lr][(]u[)]r[(]E[)]     printf("%c%c%c",yytext[0],yytext[1], yytext[2]);
[lm][(]u[)]r[(]E[)]             printf("%cE",yytext[0]);
RR[(]u[)]r[(]E[)]               printf("rrE");

        /* Hiatus-Tilger*/
        /* nak-Prs: punoh(e)m  --> punohem, hap(e)m --> hapem */
        /* aor: puno(v)a  --> punova, hap(v)a --> hapa     */
        /* imf:  puno(n)te --> punonte  hap(n)te --> hapte  */
        /* imv: Shkrua(j) --> shkruaj, shkrua(j)e --> shkruaje */
        /* shkrua(j)mE --> shkruamE */
        /* opt: rro(f)tE --> rroftE, hap(f)tE --> haptE */

{VOKAL}[(][fjv][)]/("[")?({VOKAL}|[\n])     printf("%c%c",yytext[0], yytext[2]);
{VOKAL}[(][h][)]/{VOKAL}                    printf("%c%c",yytext[0], yytext[2]);
{VOKAL}[(][fn][)]                           printf("%c%c",yytext[0], yytext[2]);
```

```
[(][fhjnv][)]                          ;

        /* prs [13]pl: punoj-(i)mi(E) --> punojm, hap-(i)m(E) --> hapim */
        /*  bie(i)m(E)        --> biem  */
        /* aor [1-3]pl: vra-(E)m(E) --> vramE,  hap-(E)m(E) --> hapEm */
        /* punua(E)m(E) -->  punuam */
        /* par: punua(u)r(E) --> punuar, pi(u)r(E) --> pirE      */
        /* hap(u)r(E) --> hapur                                  */

{DVOKAL}[(][Eiu][)][mntr][(]E[)]    printf("%c%c%c",yytext[0],yytext[1],
                                    yytext[5]);
{JVOKAL}[(][Eiu][)][mntr]([(]E[)])      printf("%c%c%c",yytext[0],yytext[4],
                                        yytext[6]);
[(][Eiu][)][mntr]([(]E[)])?             printf("%c%c", yytext[1], yytext[3]);
[(][i][)]                               printf("%c", yytext[1]);

        /* aor 3sg:  ik(i|u) --> iku hyr(i|u) --> hyri     */
        /* pii(i|u) --> piu, mEso(i|u) --> mEsoi i         */

[std]h[(]i[|]u[)]                   printf("%chi",yytext[0]);
{IK}[(]i[|]u[)]                     printf("%cu",yytext[0]);
[(]i[|]u[)]                         printf("i");

        /* kon prs 2sg: puno(E)sh --> punosh,  hap(E)sh --> hapEsh */

{VOKAL}[(]E[)]                      printf("%c",yytext[0]);
[(]E[)]                             printf("E");



        /* Phonologisch bedingter Stammwechsel:*/
        /*  maRR  --> marr, moRRa --> mora */

oRR                                 printf("or");
RR                                  printf("rr");

        /* sJell --> sjell, sJillni --> sillni, sJolla  --> solla */

J/[iou]                             ;
J                                   printf("j");

        /* pJek --> pjek pJiKni --> piqni  pJoKa --> poqa    */

[io]K                               printf("%cq",yytext[0]);
K                                   printf("k");


        /* dJeG --> djeg, diGni --> digjni, doGa --> dogja    */

[io]G                               printf("%cgj",yytext[0]);
G                                   printf("g");
```

```
        /* Klitik-Gruppe */

"["                                 {BEGIN KLI;}
"]"                                 {BEGIN 0;}

        /* mE + u --> m'u, tE + ju --> t'ju, tE + i --> t'i */

<KLI>E[ ]/[iju]                     printf("'");

        /* mE + e  --> ma */

<KLI>E[ ]e                          printf("a");

        /* u + i --> ua */

<KLI>[iu][ ][ei]                    printf("%ca",yytext[0]);

        /* ju +u --> ju u */

<KLI>[j][u][ ]u                     ECHO;

        /* iu +u --> iu */
<KLI>[iu][ ][u]                     printf("iu");

        /* puno(j) +  i + u --> punoju */

{VOKAL}[(]j[)]"["[iu][ ]u   printf("%cju",yytext[0]);

        /* puno(j) +  i + e --> punoja */

{LAUT}([(]j[)])?"["[i][ ][ei]       printf("%cja",yytext[0]);

        /* Defaults */

[\n]                                |
.                                   ECHO;
```

B.2.4 mo.m

```
%start ADM AOR AO2 AO3 AO4 AOX HIP IMF HI2 IM2 JAM KAM KLIN KLI2 KOL KOKL KON
        KONN KO2 NAK NA2 NOR OPT OP2 PRS VETE
%%

        /* ****************** PRAEVERBALE PARTIKEL ************************ */

[(]fut[)]                   -->     do
[(]jus[)]                   -->     le
[(]ger[)]                   -->     duke    {START KONN;}
[(]priv[)]                  -->     pa      {START KONN;}
[(]inf[)]                   -->     pEr     {START KONN;}
[(]abs[)]                   -->     me
[(]kon[)][ ]/[(][adn]       -->     [tE     {ELSE KON;}
[(]kon[)]                   -->     tE      {STAY KONN; ELSE KON;}
```

```
/* ***************** FUELLMATERIAL ****************************** */

<KON,KOL,KOKL>[(]/[adn]                 -->     " "
[(]/[adn]                               -->     [       {MERGE KOKL KONN; ELSE KOL;}
[(]/S                                   -->     ;       {STAY ADM KLIN KON; ELSE NOR;}

        /* ***************** KLITIKA ************************************ */

kli             -->     ]       {MERGE KONN KOKL; STAY KLIN KON; ELSE  NOR;}
[ad]1s          -->     mE
[ad]2s          -->     tE
a3s             -->     e
(d3s|a3p)       -->     i
[ad]1p          -->     na
[ad]2p          -->     ju
d3p             -->     u
nak/[(]kli      -->     u       {START KLIN;}

        /* *************** STAEMME ************************************ */

&KONTEXT:       "([+]V[123][0-9a-z])?([+](K[123][0-9a-z]|[a-zA-Z]{1,3}))?[)]"

        /* punoj */
S1                      -->     pun
        /* shkruaj */
S2                      -->     shkr
        /* ruaj */
S3                      -->     r
        /* luaj */
S4                      -->     l
        /* kthej */
S5                      -->     kth
        /* lyej */
S6                      -->     l
        /* zhyej */
S7                      -->     zh
        /* zjej */
S8                      -->     zje
        /* laj */
S9                      -->     la
        /* fshij */
S10                     -->     fshi
        /* brej */
S11                     -->     bre
        /* rroj */
S12                     -->     rro
        /* mbroj */
S13                     -->     mbro
        /* yej */
S14                     -->     ;
        /* mbaj */
S15                     -->     mba
        /* bEzaj */
S16                     -->     bEza
        /* buj */
S17                     -->     bu
        /* arrij */
```

```
S18                     -->     arri
        /* shty */
S19                     -->     shty
        /* bEj */
S20                     -->     bE
        /* blej */
S21                     -->     bl
        /* ndiej */
S22                     -->     nd
        /* mErdhij */
S23/(aor|opt|par)       -->     mardh
S23                     -->     mErdhi
        /* dErsij */
S24/(aor|opt|par)       -->     dirs
S24                     -->     dErsi
        /* hyj */
S25                     -->     hy
        /* prij */
S26                     -->     pri
        /* hap   */
S27                     -->     hap
        /* dElir   */
S28                     -->     dElir
        /* prish   */
S29                     -->     prish
        /* rrok   */
S30                     -->     rrok
        /* hipi   */
S31/[1-3]sg             -->     hip  {MERGE HIP NOR; MERGE HI2 KON;ELSE &NORM;}
S31                     -->     hip
        /* heq   */
S32                     -->     h
        /* sjell   */
S33                     -->     sJ
        /* vjel   */
S34                     -->     vJ
        /* vjedh   */
S35                     -->     vJ
        /* dridh   */
S36                     -->     dr
        /* prier   */
S37                     -->     pr
        /* fsheh   */
S38                     -->     fsh
        /* ndez   */
S39                     -->     nd
        /* ngjesh   */
S40                     -->     ngj
        /* rrah   */
S41                     -->     rr
        /* njoh   */
S42                     -->     nj
        /* nxjerr   */
S43                     -->     nxJ
```

```
        /* pjek   */
S44                     -->     pJ
        /* djeg   */
S45                     -->     dJ
        /* dal   */
S46                     -->     d
        /* marr   */
S47                     -->     m
        /* stErvit   */
S48                     -->     stErvi
        /* vendos   */
S49                     -->     vendos
        /* shetit   */
S50                     -->     sheti
        /* pyes   */
S51                     -->     pye
        /* spErkas   */
S52                     -->     spErk
        /* trokas   */
S53                     -->     trok
        /* humbas   */
S54                     -->     humb

        /************* STAEMME MIT VOKALERSCHIEBUNG ****************** */

        /* vErras   */
S55/([1-3](sg|pl)|imf)          -->     vErr
S55                             -->     virr
        /* thErras   */
S56/([1-3](sg|pl)|imf)          -->     thErr
S56                             -->     thirr
        /* shtErras   */
S57/([1-3](sg|pl)|imf)          -->     shtErr
S57                             -->     shterr
        /* kEllas   */
S58/([1-3](sg|pl)|imf)          -->     kEll
S58                             -->     kall
        /* kElthas   */
S59/([1-3](sg|pl)|imf)          -->     kElth
S59                             -->     klith
        /* bErtas   */
S60/([1-3](sg|pl)|imf)          -->     bErt
S60                             -->     brit
        /* pErkas   */
S61/([1-3](sg|pl)|imf)          -->     pErk
S61                             -->     prek
        /* kErcas   */
S62/([1-3](sg|pl)|imf)          -->     kErc
S62                             -->     kris
        /* pElcas   */
S63/([1-3](sg|pl)|imf)          -->     pElc
S63                             -->     plas
        /* flas   */
S64/([1-3](sg|pl)|imf|nak)      -->     fl
S64                             -->     fol
```

```
        /* shklas   */
S65/([1-3](sg|pl)|imf)          -->     shkl
S65                             -->     shkel

        /************** STAEMME MIT VOKALERSCHIEBUNG (ENDE) *************** */

        /* vras   */
S66                             -->     vr
        /* ngas   */
S67                             -->     ng
        /* gErgas   */
S68                             -->     gErg
        /* shkas   */
S69                             -->     shk
        /* pres   */
S70                             -->     pr
        /* dhjes   */
S71                             -->     dhJ
        /* pres2   */
S72                             -->     pr
        /* vdes   */
S73                             -->     vd
        /* loz   */
S74                             -->     lo
        /* pi   */
S75                             -->     pi
        /* di   */
S76                             -->     di
        /* ngre   */
S77                             -->     ngr
        /* fle   */
S78/(aor|opt|par)               -->     f
S78                             -->     fl
        /* zE   */
S79                             -->     z
        /* vE   */
S80                             -->     v
        /* shpie   */
S81                             -->     shp
        /* shtie   */
S82                             -->     sht
        /* lE   */
S83/aor                         -->     l       {START AO3;}
S83                             -->     l
        /* dua   */
S84                             -->     d

        /* ************ STAMMALTERNATION ***************** */

        /* them   */
S85/[1-3](sg|pl)                -->     th      {MERGE KO2 KON; ELSE JAM;}
S85/aor                         -->     th      {START AO3;}
S85/imf                         -->     th      {MERGE JAM NOR KON;}
S85                             -->     th
```

```
        /* vete   */
<NOR>S86/[2-3]sg                --> v       {START VETE;}
S86/1sg                         --> v       {START VETE;}
S86/1pl                         --> v       {START JAM;}
S86                             --> v
        /* gjej   */
S87                             --> gj
        /* lind   */
S88                             --> lind
        /* bie   */
S89/(aor|opt|par)               --> pr
S89                             --> b
        /* bie2   */
S90/aor                         --> r       {START AO3;}
S90/(opt|par)                   --> r       {START AO3;}
S90                             --> b
        /* ha   */
S91/(opt|par)                   --> ngrEn
<KLIN>S91/aor[)]3sg             --> hangEr
S91/aor                         --> hangr
S91                             --> ha
        /* rri   */
S92/(aor|opt|par)               --> nde
S92                             --> rri
        /* shoh   */
S93/(opt|par)                   --> p
S93/aor                         --> p       {START AO3;}
S93                             --> sh
        /* vij   */
S94/(imv|aor|opt|par)           --> ;
<NOR>S94/[23]sg                 --> vj
S94                             --> v
        /* jap     */
S95/aor                         --> dh      {START AO3;}
S95/(opt|par)                   --> dh
S95                             --> j

        /* jam  */
S96/aor                 --> q               {START AO3;}
S96/(opt|par)           --> q
S96/3sg                 --> ;               {MERGE JAM NOR;}
S96/imf                 --> ;               {MERGE JAM NOR KON;}
S96                     --> j               {MERGE KO2 KON; ELSE JAM;}
        /* kam  */
<KLIN>S97/aor[)]3sg     --> p
S97/aor                 --> p               {START AO4;}
S97/opt                 --> p
S97/par                 --> p
S97/(3pl|imf)           --> k       { STAY ADM; MERGE KO2 KON; ELSE KAM;}
S97                     --> k       {MERGE KO2 KON; ELSE KAM;}

        /*  ****************** VOKALE ******************************** */

&KONTEXT:       "[+](K[123][0-9a-z]|[a-zA-Z]{1,3})[)]"
```

```
        /* ************* VOKALE 3. KONJUGATION ********************** */

        /*  Y+O */
V34/(aor|opt|par)                                   -->     y
V3c/opt                                             -->     o
        /* E: */
V2p/(opt|par)                                       -->     E
V3[37b]/(opt|par)                                   -->     E
<JAM>V3c/3sg                                        -->     E
V3[cd]/imf                                          -->     E       {MERGE KAM ADM;}
        /* U */
V3a/(opt|par)                                       -->     u
V3[67]/nak                                          -->     u
V3[3a]/aor                                          -->     u
V2n/aor                                             -->     u
        /* UA */
V37/2sg                                             -->     ua
V3[7]/imv[)](2sg|[(][ad]3[sp])                      -->     ua
V36/(1sg|[13]pl)                                    -->     ua
V3[6]/imv                                           -->     ua
<KON>V36/[2sg]                                      -->     ua
        /* JE */
<KON>V3[34ab]/3sg                                   -->     je
V3[34ab]/imv                                        -->     je
V31/(aor|opt)                                       -->     je
V2z/(aor|opt|par)                                   -->     je
        /* IE */
V2j/par                                             -->     ie
<KLIN>V2j/aor[)]3sg                                 -->     ie
        /* I  */
V38/imf                                             -->     i
V3[cd]/(imf|imv|nak)                                -->     i
<KLIN>V31/aor[)]3sg                                 -->     i
V31/nak                                             -->     i
        /* E  */
V37/1(sg|pl)                                        -->     e
V36/aor                                             -->     e
V3k/(imv|aor)                                       -->     e
        /* A       */
V3[57bo]/aor                                        -->     a
V2o/aor                                             -->     a
V3[6dk]/(opt|par)                                   -->     a
V2o/(opt|par)                                       -->     a
V38/(opt|par|aor)                                   -->     a
<AO4,KLIN>V3d                                       -->     a
<JAM,KAM,ADM,NOR>V3[cd]/([13]sg|3pl)                -->     a

        /* *********** VOKALE 1. KONJUGATION ********************** */

        /* O-UA-Gruppe */
<KLIN>V1[01]/aor[)]3sg                              -->     ua
<KLIN>V2q/aor[)]3sg                                 -->     ua
V1[01]/aor[)][1-3]pl                                -->     ua
V10/par                                             -->     ua
V1[01]/(aor|opt)                                    -->     o
V10                                                 -->     o
V13/aor                                             -->     o
```

```
V13/par[)][ \n \t]                          -->     o
V1[1-3]/nak                                 -->     u
V1[1-3]                                     -->     ua
        /* E-YE-Gruppe */
<KLIN>V1[456]/aor[)]3sg                     -->     ye
V1[456]/aor[)][1-3]pl                       -->     ye
V1[45]/par                                  -->     ye
V16/par                                     -->     y
V1[456]/(aor|opt)                           -->     e
V1[56a]/nak                                 -->     y
V14                                         -->     e
V1[56a]                                     -->     ye

        /* ************* VOKALE 2. KONJUGATION ******************** */

        /* E */

        /* E IN 2/3SG */
<NOR>V2[5679a-dfghop]/[23]sg                -->     e
<NOR>V3k/[23]sg                             -->     e
        /* E IN 2PL/IMF...  */
V2[7p]/(nak|imf|imv|2pl)                    -->     e
        /* E BEI AOR 2SG */
V2[op]/aor[)]2sg                            -->     e
V3[57b]/aor[)]2sg                           -->     e
        /* O IM AORIST */
V2[1237dq]/aor                              -->     o
        /* I */
V2[l]/(2pl|imf|nak|imv|aor)                                 -->     i
V2[9hkm]/(2pl|imf|nak|imv|aor|opt|par)                      -->     i
V2a/(2pl|imf|nak|imv|aor|opt)                               -->     i
V2[13456dqhjoz]/(2pl|imf|nak|imv)                   -->     i
V30/(2pl|imf|nak|imv)                                       -->     i
V2[imn]/(2pl|imf|nak)                                       -->     i
V3[345ab]/(2pl|imf|nak)                                     -->     i
V2[bcfg]/(2pl|imf)                                          -->     i
V2[fgn]/nak                                                 -->     i
V39/nak                                                     -->     i
V22                                                         -->     i
V3k                                                         -->     i

        /* STAMMVERKUERZUNG */

V2[bcf]/(nak|imv|aor|opt|par)                               -->     ;


        /* ************** Defaults ******************** */

        /* E-STAEMME */
V2[14ciqjklmz]          -->     e
V3[089cd]               -->     e
        /* IE-STAEMME */
V23                     -->     ie
V3[134ab]               -->     ie
```

```
        /* A-STAEMME */
V2[579abfdpgh]          -->     a
        /* O-STAEMME */
V2[6o]                  -->     o
V3[67]                  -->     o
        /* E:-STAEMME */
2n                      -->     E
V35                     -->     E

        /* ************ KONSONANTEN   ****************** */

&KONTEXT:       "[)]"

        /* EINZELNE MUSTER */

        /* JAM + KAM  */
K3e/opt                         -->     C
K3e/par                         -->     s
        /* VIJ */
K1e/(opt|aor|par)               -->     rdh
K1e/imv                         -->     ja
        /* LOZ */
K34/(1sg|[13]pl|nak)            -->     z
<KON>K34/[23]sg                 -->     z
        /* VDES */
K29/(aor|opt|par)               -->     K
        /* SH */
<JAM>K3d/3sg                    -->     sh
K1a/(aor|opt|par)               -->     sh
        /* J(T|C) */
K1c/opt[)](1-3]pl|[12]sg)       -->     jC
K13/(aor|par)                   -->     jt
K3[4a]/(aor|par)                -->     jt
K1[2c]/(aor|opt|par)            -->     jt
K14/(aor|opt|par|nak)           -->     jt
        /* N(J|D) */
<NOR>K19/aor[)]3sg              -->     n
K19/aor[)][1-2]sg       -->     n
K3[578]/(opt|par)               -->     n
<NOR>K1[0-9bde]/[23]sg          -->     n
K3a/opt[)]3sg                   -->     n
K3a/opt                         -->     nj
K3c/(opt|par)                   -->     n
K3d/par                         -->     n
K1d/nak                         -->     nd
        /* T */
K1[bd]/(aor|opt|par)            -->     t
K2a/(aor|opt|par)               -->     t
K15/(imv|aor|opt|par)           -->     t
K3e/aor                         -->     t
K34/[23]sg                      -->     t
        /* R */
<KON>K17/3sg                    -->     r
<KON>K3[567]/3sg                -->     r
K18/imv                 -->     r
K3[567]/imv                     -->     r
<NOR>K1[68]/aor[)]3sg           -->     r
```

```
<NOR>K3[56]/aor[)]3sg            -->    r
K1[68]/aor[)][1-2]sg             -->    r
K3[56]/aor[)][1-2]sg             -->    r
K36/nak                          -->    r
        /* STANDARDS FUER 2. KONJUGATION */
K1[0-9bde]/(1sg|[13]pl)                 -->    j
K1[0-79be]/imv[)][(]nak                 -->    h
K1[0-79be]/imv[)][(][ad][1-3][sp]       -->    (j)
K1[1246ad]/imv                          -->    (j)
K3[48]/imv                              -->    (j)

        /* ************ KONSONANTEN  2. KONJUGATION ****************** */

        /* STAMMVERKUERZUNG */
K2[467]/(aor|opt|par)            -->    ;
K3b/(aor|opt|par)                -->    ;
K3c/aor                          -->    ;
K2[46]/imv                       -->    ;
K24/nak                          -->    ;
        /* T > S */
<KON>K2[234678]/[23]sg           -->    s
K2[234678]/(1sg|[13]pl)          -->    s
K2[3]/imf                        -->    s
K2[1234678]/imf[)][1-3]pl        -->    s
K2[02134678]/imf[)]3sg           -->    s
K2[01234678]                     -->    t

        /* ************ KONSONANTEN -DEFAULTS ******************** */

K1[0-9a-e]                       -->    ;
K2a                              -->    ;
K3[45678ade]                     -->    ;
K29                              -->    s
K3c                              -->    p
K3b                              -->    h

        /* ************** INFL- MORPHEME ********************** */

<NAK>imf/3sg        -->    j       {START NOR;}
imf/[1-3](sg|pl)    -->    sh      {MERGE IM2 JAM KAM NAK KO2 ADM;}
imf/[1-3]pl         -->    n       {START IMF;}
imf                 -->    ;       {START IMF;}
aor/3sg             -->    ;       {STAY KLIN AO3; ELSE AOR;}
aor                 -->    ;       {STAY AO3; ELSE AOR;}
opt/3sg             -->    ;       {START OPT;}
opt                 -->    sh      {START OPT;}
imv                 -->    ;       {START NOR;}
nak                 -->    e       {START NAK;}
par/[ ][(][([(])?S  -->    E       {START 0;}
par/[(][(][([(])?S  -->    ;       {START ADM;}
par                 -->    (u)r(E)
```

```
        /* *************** FUELLMATERIAL ********************** */

&KONTEXT:     NULL

[)]/1(sg|pl)                --->    ;       {MERGE NAK JAM KAM KO2;}
[)]/nak                     --->    (h)
<AOR,AO3>[)]/[1-3]pl        --->    (E)
<PRS,AO2>[)]                --->    (E)     {START 0;}
<NA2>[)]                    --->    i       {START 0;}
<OPT>[)]/[1-3]pl            --->    i       {START OP2;}
<IMF>[)]/[12]sg             --->    j
[)]/[1-3]pl                 --->    i       {MERGE IMF IMF IM2;}
<AOR>[)]/[12]sg             --->    (v)
[)]/opt                     --->    (f)
[)]/[13]pl                  --->    (i)     {MERGE PRS NOR KON KO2 JAM KAM HIP;}

        /* ***************** ENDUNGEN ************************ */

        /* Optativ */
<OP2>2pl                    --->    ;
        /* Imperfekt- und Aorist */
<IM2,IMF,AOR,OPT>1sg        --->    a
<IM2,IMF,AOR>2sg            --->    e
<IM2,IMF>3sg                --->    (n)te
2pl                         --->    t       {STAY IMF; MERGE AO2 AOR AO3;}
<AOR>3sg                    --->    (i|u)
        /* Praesens Nichtaktiv */
<NAK>1sg                    --->    m
<KON,KO2,NAK,HI2>2sg        --->    (E)sh
<NAK>3sg                    --->    t
<KON,HI2>3sg                --->    (j)E
        /* Irregulaeres Praesens */
<VETE>[1-3]sg               --->    te
<HIP,HI2>1sg                --->    i
<HIP>[23]sg                 --->    En
<JAM,KO2,OPT>3sg            --->    tE
        /* Irregulaerer Aorist */
<AO3>1sg        --->    shE
        /* Defaults */
[1-3]sg         --->    ;
1pl             --->    m               {MERGE NA2 NAK;
                                MERGE AO2 AOR AO3; ELSE &NORM;}
2pl             --->    ni
3pl             --->    n               {MERGE AO2 AOR AO3; ELSE &NORM;}
[ ]             --->    " "
[a-zA-Z] --->   ECHO;
.               --->    ;
```

B.2.5 phoen.l

```
%{
#include <string.h>
int count = 1;                  /* Zeilenzaehler */
char muster[80];                /* Speicher fuer Muster */
char kontext[100];              /* Speicher fuer Kontexte */
char string[80];                /* Speicher fuer String */
char zustand1[20];              /* 1. Argument fuer Automaten-Befehle */
char zustand2[20];              /* 2. Argument fuer Automaten-Befehle */
%}
%start  PFEIL STRING PSTRING BEFEHLE MER1 MER2 MER3 START1 START2 STAY1 STAY2
        KONTEXT
%%

        /* ****************  KOMMENTARE  ************************** */

^[\t ].*\n                      |
^[\n]                           {ECHO; ++count; BEGIN 0;}


        /* ****************  KONTEXTE        ************************** */

^&KONTEXT[:]                    {BEGIN KONTEXT; }
<KONTEXT>["][^"\n]*["]          {q_copy(kontext,yytext, yyleng);
                            printf("\n"); BEGIN 0;}
<KONTEXT>NULL                   {strcpy(kontext,""); printf("\n");BEGIN 0;}
<KONTEXT>[^ \t\n]               {f_meld(1,count); exit(1);}

        /* ****************  FLEX-HEADER ************************* */

[%].*\n                         {ECHO; ++count;}

        /* ****************  AUTOMATENBEFEHLE    ****************** */

<PSTRING>[\n]                   {onlyprint();  ++count; BEGIN 0;}
<PSTRING>[{]                    {BEGIN BEFEHLE;}
<BEFEHLE>[}]                    {BEGIN 0;}
<BEFEHLE>MERGE                  {BEGIN MER1;}
<BEFEHLE>(START|ELSE)           {BEGIN START1;}
<BEFEHLE>STAY                   {BEGIN STAY1;}
<BEFEHLE>[^\t\n]                {f_meld(6,count); exit(1);}

        /* ****************  MERGE   ************************** */

<MER1>[A-Z0-9]+                 {strcpy(zustand1,yytext); BEGIN MER2;}
<MER2,MER3>[A-Z0-9]+            {strcpy(zustand2,yytext);mergeprint(); BEGIN MER3;}
<MER3>[;]                       {BEGIN BEFEHLE;}
<MER1,MER2>[^ \t]               {f_meld(2,count); exit(1);}

        /* ****************  START/ELSE      ****************** */

<START1>&NORM[;]                {onlyprint(); BEGIN BEFEHLE;}
<START1>[A-Z0-9]+               {strcpy(zustand1,yytext); BEGIN START2;}
<START2>[;]                     {startprint();              BEGIN BEFEHLE;}
<START1,START2>[^ \t]           {f_meld(3,count); exit(1);}
```

```
        /* ****************  STAY      ************************** */

<STAY1>[A-Z0-9]+                {strcpy(zustand1,yytext);
                        strcpy(zustand2, yytext); BEGIN STAY2;}
<STAY2>[A-Z0-9]+                {mergeprint(); strcpy(zustand1,yytext);
                        strcpy(zustand2, yytext);}
<STAY2>[;]                      {mergeprint();  BEGIN BEFEHLE;}
<STAY1,STAY2>[^\t]              {f_meld(4,count); exit(1);}

        /* ****************  MUSTER  ************************** */


^[^\t %&\n]+([ ][^\t ]+)*    {strcpy(muster,yytext); BEGIN PFEIL;}

        /* ****************  PFEIL     ************************** */

<PFEIL>[\t ]+"-->"[\t ]+         {BEGIN STRING;}

        /* ****************  STRINGS  ************************** */

<STRING>[;]                     {strcpy(string,""); BEGIN PSTRING;}
<STRING>["][^\n]+["]            {q_copy(string,yytext,yyleng); BEGIN PSTRING;}
<STRING>[^\t \n]+               {strcpy(string,yytext); BEGIN PSTRING;}
<STRING>[\n]                    {f_meld(5,count); exit(1);}


        /* ****************  DEFAULTS ***************** ********* */

[\n]                            {++count;}
[ \t]                           |
.                               {;}
%%

        /* ****************  ONLYPRINT ************************** */


onlyprint()                     /* drucke Anweisungen ohne Kontexte */
{
musterprint();
stringprint();
printf("}\n");
}

        /* ****************  MERGEPRINT ************************** */

mergeprint()                    /* drucke MERGE-Anweisungen */
{
printf("<%s>",zustand2);
musterprint();
stringprint();
printf("BEGIN %s;",zustand1);
printf("}\n");

}
```

```
        /* ****************  STARTPRINT ************************** */
startprint()
{                                                   /* drucke BEGIN-Anweisungen */
musterprint();
stringprint();
printf("BEGIN %s;",zustand1);
printf("}\n");
}
        /* ****************  MUSTERPRINT ************************** */

musterprint()                                       /* drucke Muster */
{
        int i = 0;
if(!(strcmp(muster,"[ ]")))                         /* einzelnes Leerzeichen  -->  [ ] */
        printf("[ ]");
else
        while(muster[i] != '\0')
                {
                if(muster[i] == ' ')            /* Leerzeichen in Muster  -->  " " */
                        printf("\" \"");
                else if(muster[i] == '/')           /* Kontext einfuegen */
                        printf("/%s",kontext);
                else
                        printf("%c",muster[i]);     /* drucke alles andere  */
                ++i;
                }
}

        /* ****************  STRINGPRINT ************************** */

stringprint()                                       /* drucke Druckanweisung */
{
if(!strcmp(string,"ECHO;"))                         /* "ECHO" bleibt */
        printf("\t{ECHO;");
else
        {                                           /*  Druckanweisung */
        printf("\t{printf(\"");
        printf("%s \"); ",string);
        }
}

        /* ******************* Q_COPY ******************************** */

q_copy(char *s1, char *s2, int i)                   /* kopiert gequoteten String ohne ["] */
{
        int j;
for( j = 1; j < i-1; ++j)
        s1[j-1] = s2[j];
s1[j-1]    = '\0';
}
```

```
        /* ****************   F_MELD   ************************** */

f_meld(int i, int j)                    /* Fehlermeldungen */
{
switch(i)
        {
        case 1: fprintf(stderr,"Kontext inkorrekt Zeile %d\n",j); break;
        case 2: fprintf(stderr,"Syntax: Merge Arg1 Arg2; Zeile %d\n",j); break;
        case 3: fprintf(stderr,"Syntax: (START|ELSE) Arg; Zeile %d\n",j); break;
        case 4: fprintf(stderr,"Syntax: STAY Arg; Zeile %d\n",j); break;
        case 5: fprintf(stderr,"Syntax: Muster Pfeil String Zeile %d\n",j);                  break;
        case 6: fprintf(stderr,"Befehl-Syntax: inkorrekt Zeile %d\n",j);            break;
        }

}
```