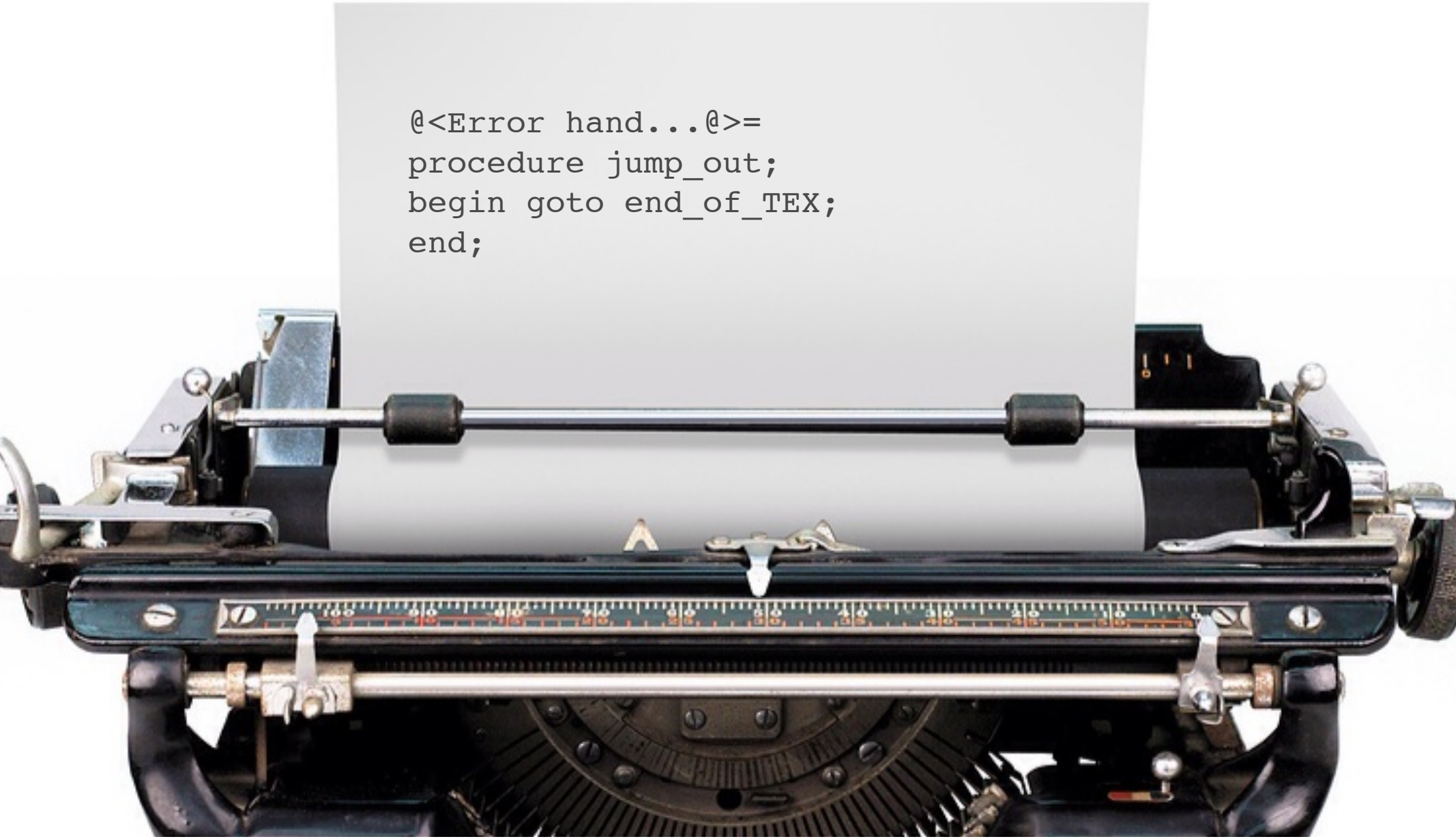


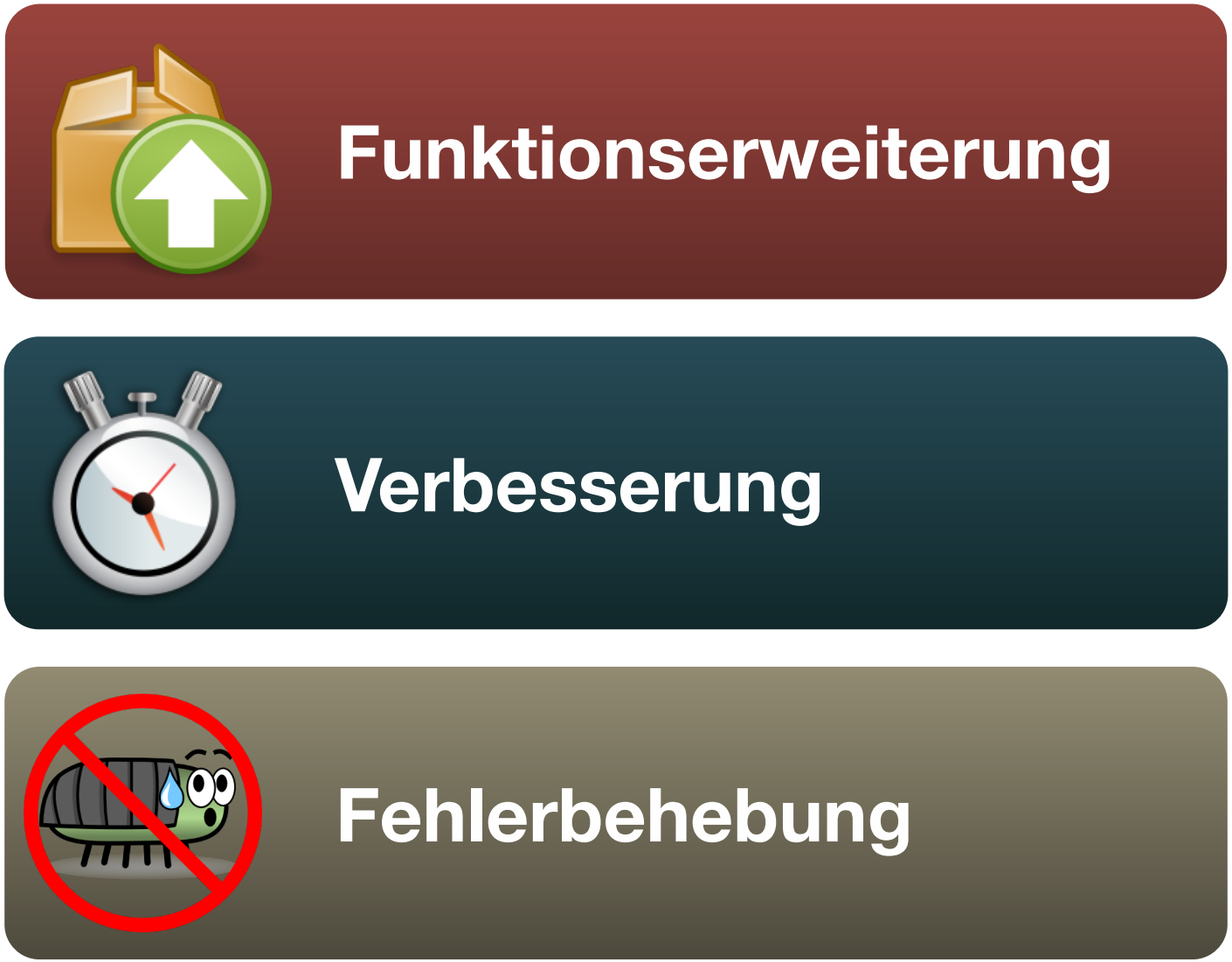
Eine historisch-kritische Betrachtung des Quelltexts von TEX

Sven Oos

Universität Trier

Köln, 01.11.2013

Softwareevolution



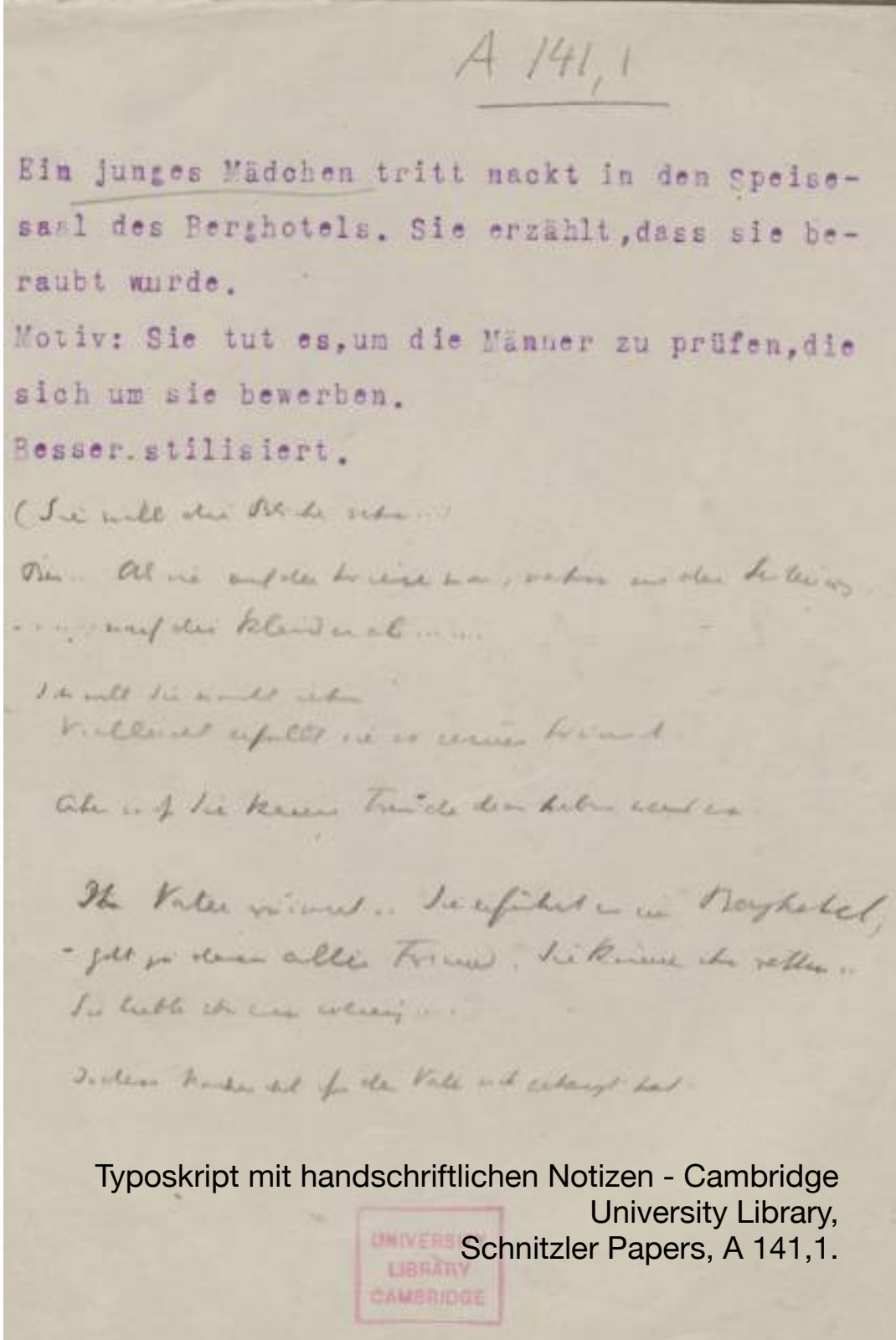
Edition

"von einem Herausgeber [...] hergestellter
und publizierter Text eines fremden Autors"

Reallexikon der deutschen Literaturwissenschaft (2007)

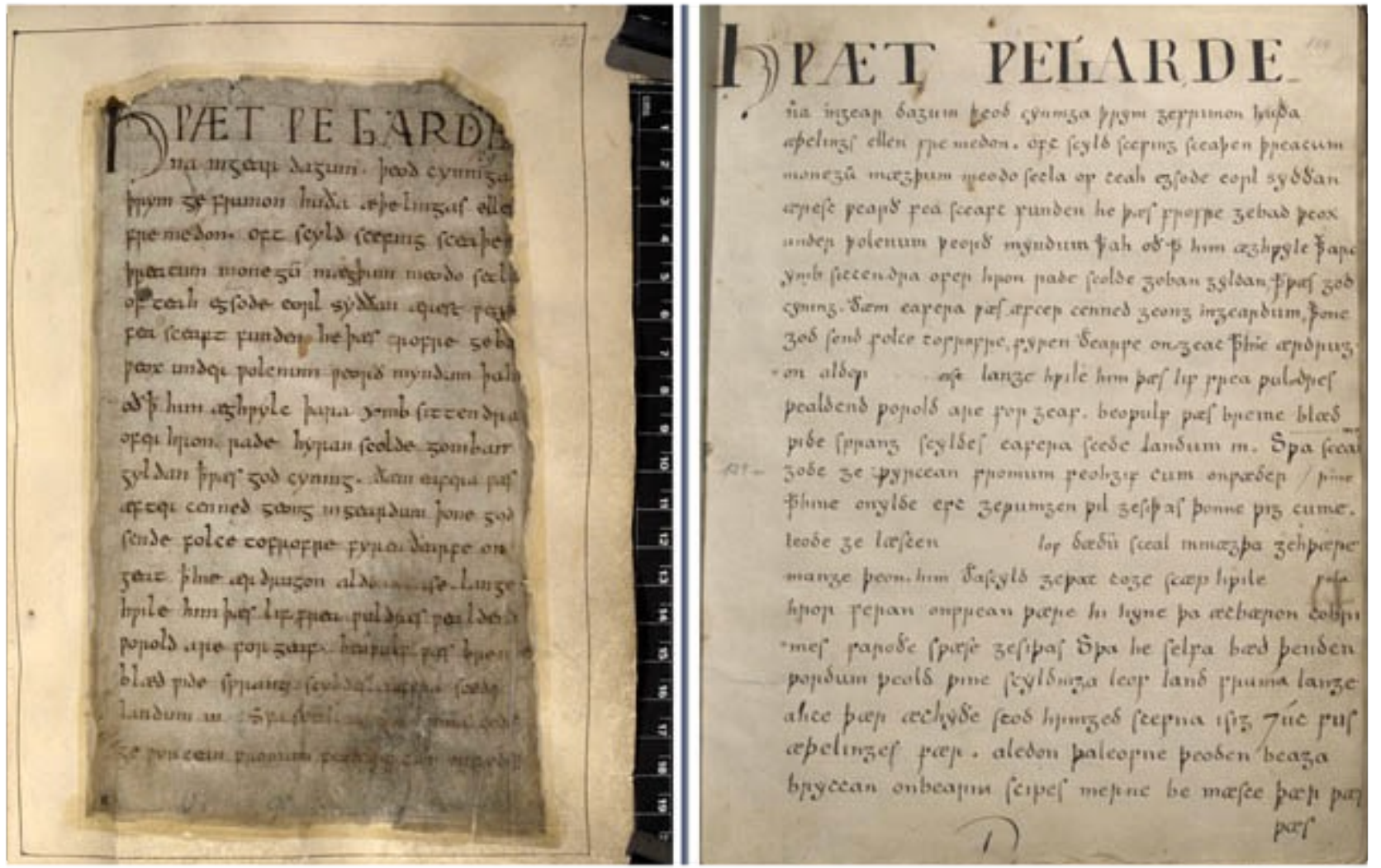
"handschriftlich oder gedruckt gelieferter Text in möglichst authentischer Form"

Reallexikon der deutschen Literaturwissenschaft (2007)



Typoskript mit handschriftlichen Notizen - Cambridge University Library, Schnitzler Papers, A 141,1.

Kollationen

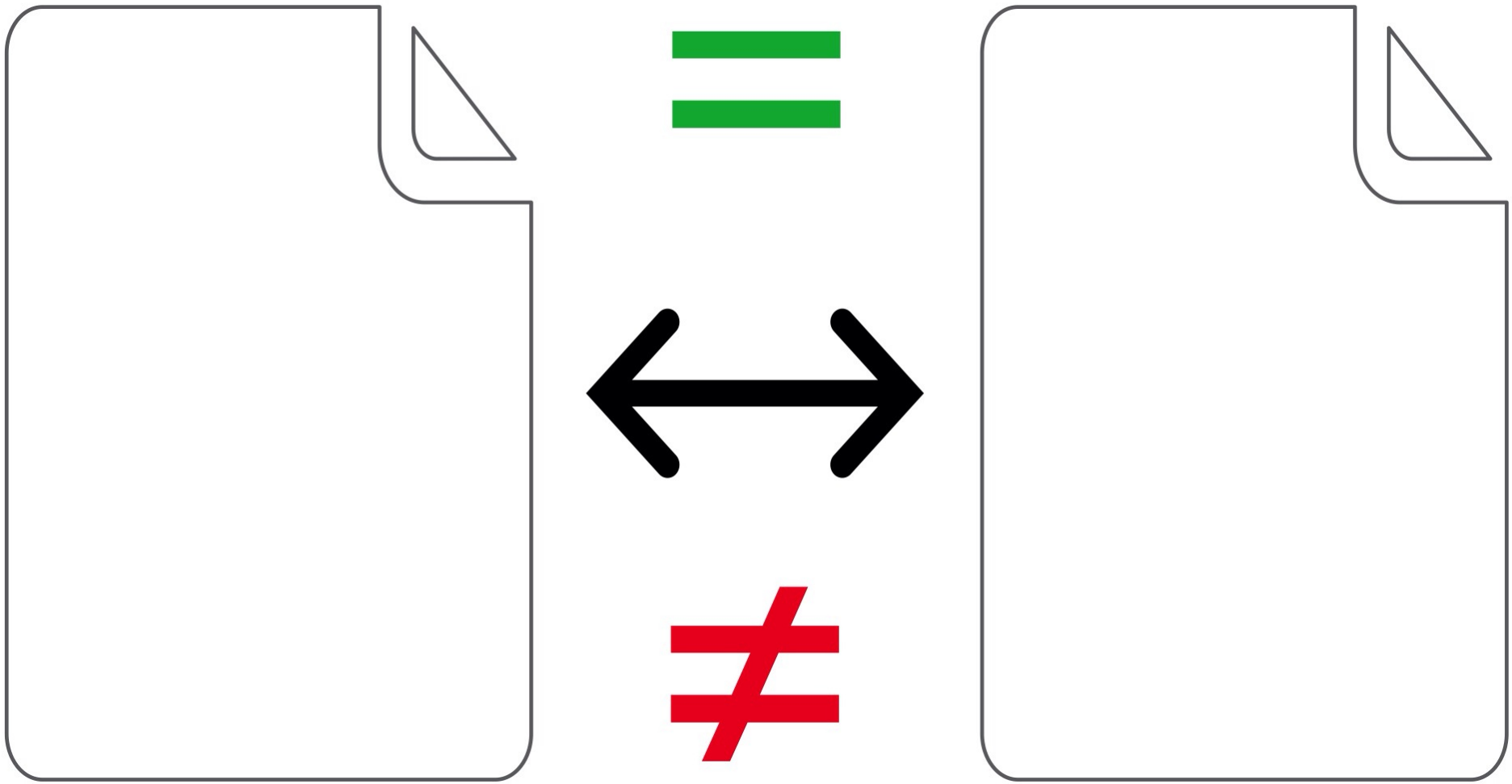


Quelle:
ebeowulf.uky.edu

Kollationen

Quelle:
ebeowulf.uky.edu

Textidentität



Textvarianz

Textkritischer Apparat (Bsp.: Treppenapparat)

Und zählen muß ich – Mit der Zahl
Schwillt immer höher meine Qual,
Mir ist als wälzten sich die Leichen
Auf meine Brust – Gottlob! sie weichen!

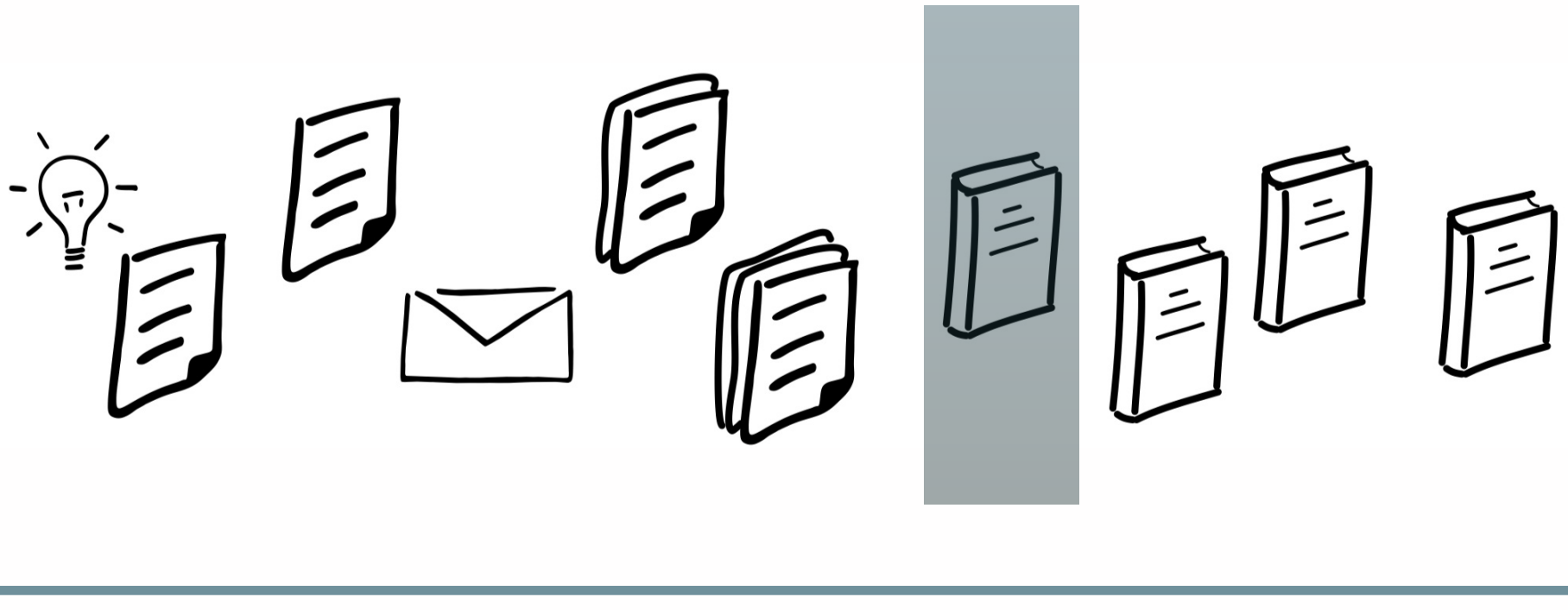
Und *bis* weichen!]

1. Ich kann nicht
2. Ich
3. Vergebens wälz ich
    a) hin u
    b) her u hin Mein armes Haupt
4. Ich [sch..] stehen immer
5. Die Nacht ist län
6. Und zählen muß ich,
7. a) ach die
    b) – mit der Zahl Schwillt immer höher meine Qual,
        i. Und auf den Kissen [hin] auf u nieder
        ii. Mir ist als lägen tausend Leichen
        iii. /: Mir ist als :/ A. wälzten /: tausend Leichen :/

B. /: wälzte :/ man die /; Leichen :/

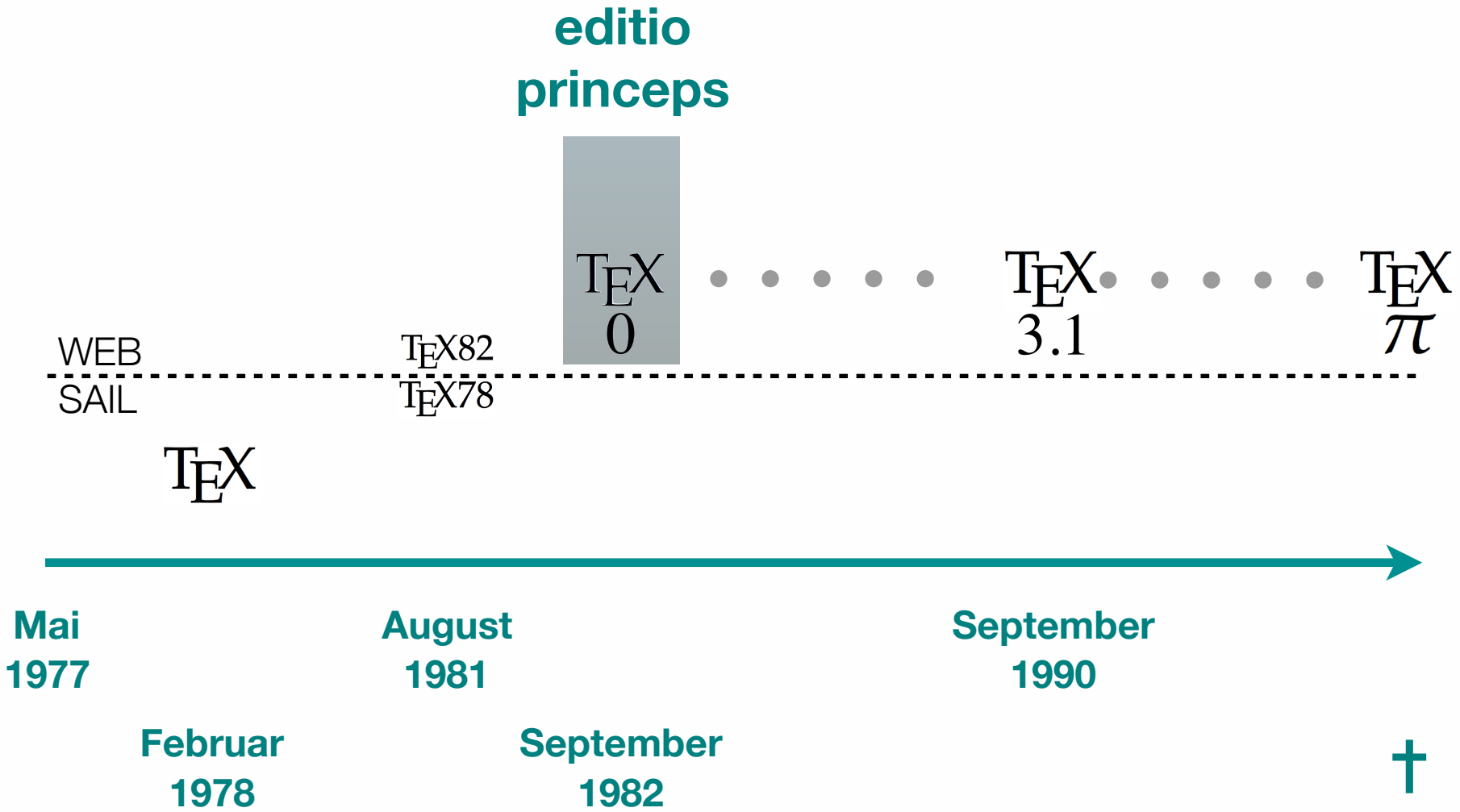
C. /: wälzten :/ sich /: die Leichen :/ Auf meine Brust – Gottlob! sie weichen! H 1

editio princeps



Idee: "Software-Edition"

- Quelltext als Klartext
- Mehrere Versionen
- 1 Programmierer


editio
princeps
TEX
0
TEX
3.1
TEX
π
WEB
SAIL
TEX82
TEX78
TEX
Mai
1977
Februar
1978
August
1981
September
1982
September
1990
†

```
internal procedure idlookup(boolean single; integer p,h) # searches the hashtable;
begin comment The pointer "p" to a packed control sequence of hash code h,
or (alternatively) the single-character code "p", is looked up in
the hash table. If not found, it is entered, and the savestack is adjusted
so that the entry will be cleared at the close of the current nesting
level. Upon exit, the appropriate index for this symbol in eqtb will appear
in the global variable "hashentry";
if single then
        begin comment single character p;
        hashentry ← p+hashsize;
        if eqtb[hashentry]≠0 then return;
        end
else    begin comment multicharacter id; integer t;
        hashentry ← hhead[h];
        while hashentry<hashsize do
                begin integer q,r; q←field(value,hash[hashentry]); r←p;
[...]
```

SAIL

```
@ Here is the subroutine that searches the hash table for an identifier
that matches a given string of length |l>1| appearing in |buffer[j..
(j+l-1)]|. If the identifier is found, the corresponding hash table address
is returned. Otherwise, if |no_new_control_sequence| is |true|, the
dummy address |undefined_control_sequence| is returned. Otherwise the
identifier is inserted into the hash table and its location is returned.

@p function id_lookup(@!j,@!l:integer):pointer; {search the hash table}
label found; {go here if you found it}
var h:integer; {hash code}
@!p:pointer; {index in |hash| array}
@!k:pointer; {index in |buffer| array}
begin @<Compute the hash code |h|@>;
p←h+hash_base; {we start searching here; note that |0≤h<hash_prime|}
loop@+begin if (text(p)>0) and (length(text(p))=l) then
                if str_eq_buf(text(p),j) then goto found;
        if next(p)=0 then
                begin if no_new_control_sequence then

[...]
```

WEB

SAILDART ORG

SAIL, the Stanford Artificial Intelligence Laboratory, and
DART, Dump and Restore Technique, tape backup utility program.

This archive presents the index to DART files recorded on tape at SAIL between November 1972 and August 1990. SAIL text files were encoded in a non-standard 7-bit character set which has been translated into UTF8 equivalents. Many SAIL binary files, such as vector plots, animation, television camera digital images, and music have been converted into HTML5 embedded formats. Demonstration of SAIL PDP-10 programs is possible using Javascript. The exact bits of each SAIL file is available in octal. Other contributions to this web site include films, scanned paper documents, and analog snapshot pictures from the period. Here is the 1972 Rolling Stone article about SAIL, a 2009 reunion group picture, links to reunion video talks given at Stanford 22 November 2009. The Celebration of John McCarthy's Accomplishments at Stanford 25 March 2012 **YouTube.com** links are listed here with text.

Published areas of this archive

#of dir	*code*	*#of filenames*	*first date*	*last date*	*description*
18	DOC	9,751	1964-01	1990-08	Documentation
6	CSR	1,735	1976-07	1983-01	Computer Science Reports
112	SYS	76,164	1964-01	1990-08	Systems
30	HE	14,090	1964-01	1986-03	Hand Eye Project
31	LSP	9,753	1965-04	1990-08	LISP programming language
14	AIL	6,017	1964-01	1982-09	SAIL programming language
4	NET	2,675	1972-05	1990-08	Network
2	3	13,982	1970-03	1990-08	System binary executables
52	MUS	10,656	1969-01	1982-01	Music
30	TEX	366	1981-02	1985-06	TeX typesetting system.
60	BGB	4,949	1970-11	1979-02	Example graduate student, myself.

Login programmer codes

code	*name*
LES	Les Earnest
JMC	John McCarthy
DEK	Donald Ervin Knuth
JC	John Chowning
LCS	Leland Smith
JAM	Andy Moorer
RWG	Bill Gosper
ME	Marty Frost
WD	Whit Diffie
DBA	Bruce Anderson
MFB	Martin Brooks

SAILDART ORG

This archive presents the inde
files were encoded in a non-s
files, such as vector plots, ani
formats. Demonstration of SA
octal. Other contributions to th
Here is the 1972 Rolling Ston
22 November 2009. The Celeb
listed here with text.

Publi

#of dir	*code*	*#of filenames*
18	DOC	9,751
6	CSR	1,735
112	SYS	76,164
30	HE	14,090
31	LSP	9,753
14	AIL	6,017
4	NET	2,675
2	3	13,982
52	MUS	10,656
30	TEX	366
60	BGB	4,949

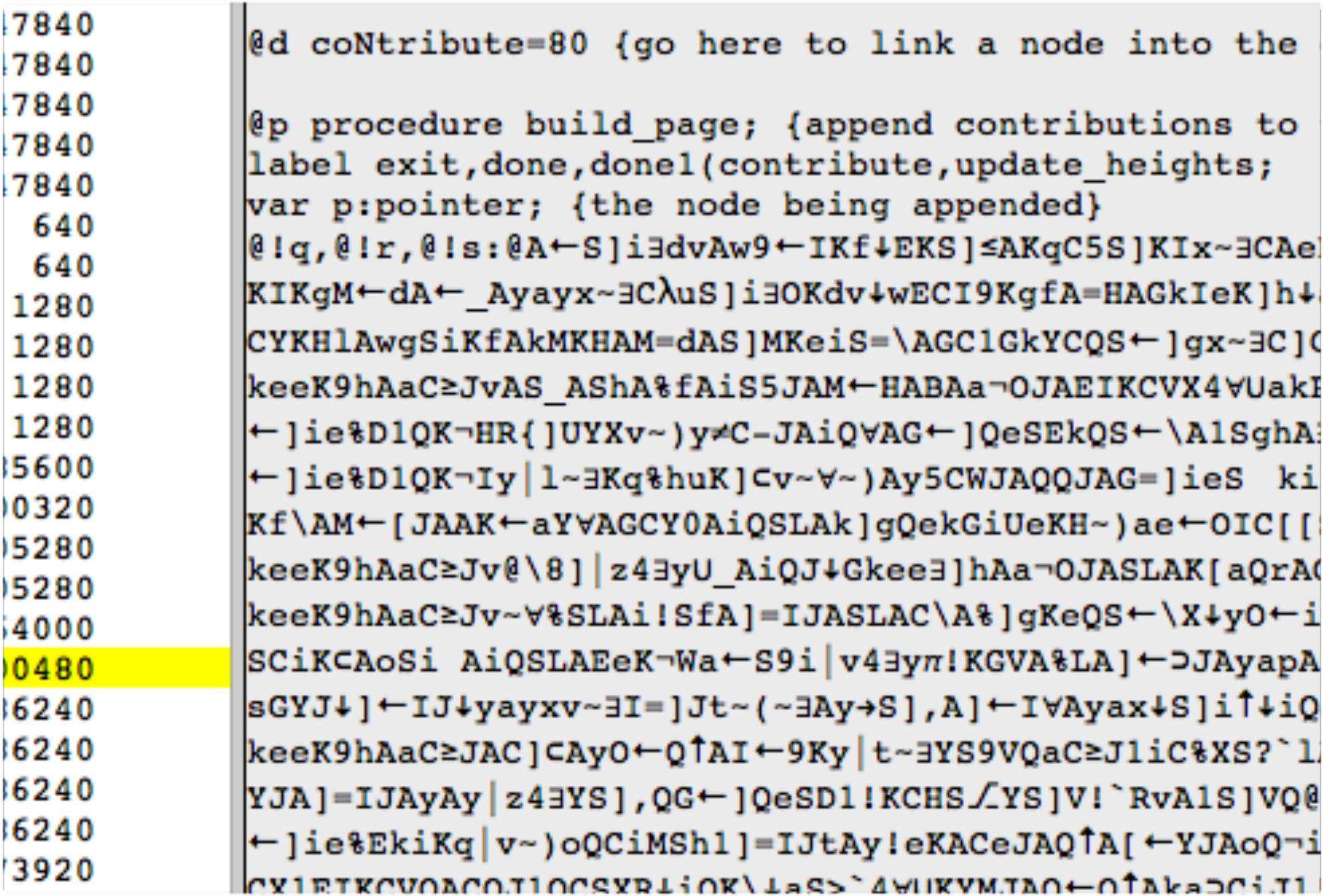
SAILDART project **TEX**, programmer **DEK** [up level] has 1970 files in 219,134,285 bytes.

buttons:[prev][next]

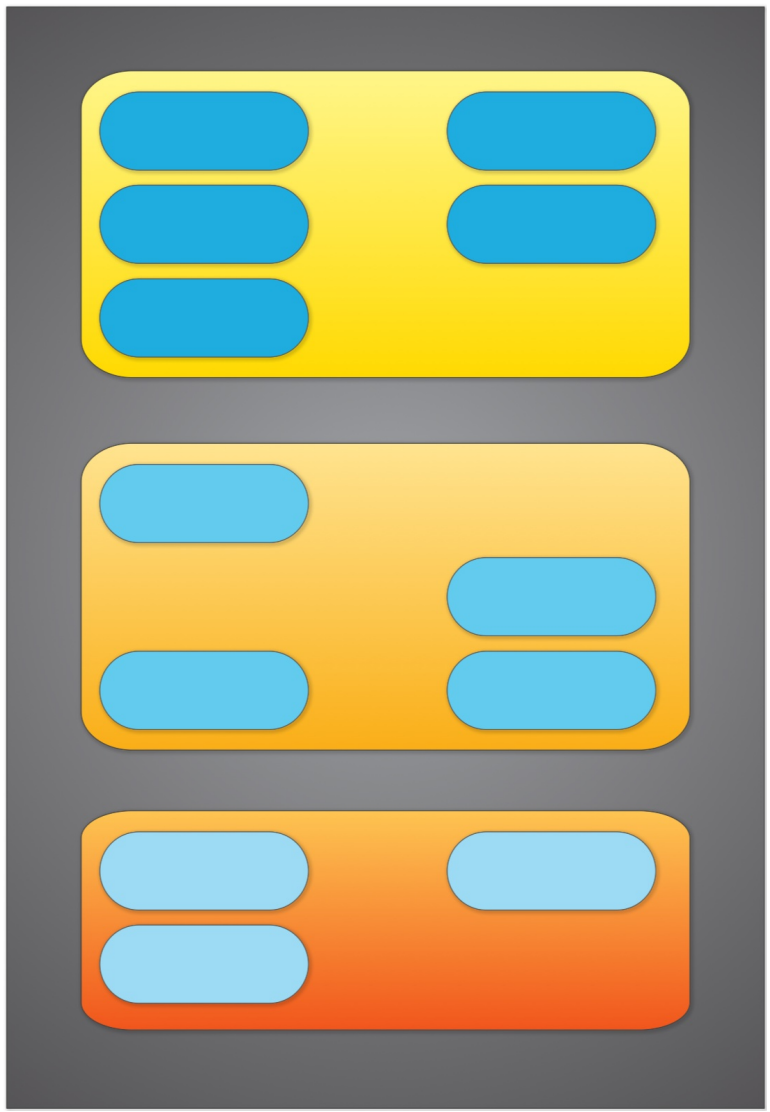
[datetime]	[Filnam.ext]	[prj,prg]	rev	[size]
1982-11-01 16:27	TEX82 .DIF	[TEX,DEK]	1	78720
1982-12-02 20:28	TEX82 .DIF	[TEX,DEK]	2	85760
1982-12-02 20:28	TEX82 .DIF	[TEX,DEK]	3	85760
1983-04-16 00:38	TEX82 .DIF	[TEX,DEK]	4	101120
1983-05-18 16:41	TEX82 .DIF	[TEX,DEK]	5	104320
1983-07-02 16:43	TEX82 .DIF	[TEX,DEK]	6	114560
1983-07-26 23:51	TEX82 .DIF	[TEX,DEK]	7	117760
1983-08-01 18:39	TEX82 .DIF	[TEX,DEK]	8	118400
1983-09-04 13:39	TEX82 .DIF	[TEX,DEK]	9	119040
1983-10-02 18:21	TEX82 .DIF	[TEX,DEK]	10	122240
1983-10-02 18:21	TEX82 .DIF	[TEX,DEK]	11	122240
1978-09-22 22:12	TEXBUG.	[TEX,DEK]	1	1920
1980-05-19 10:48	TEXCHG.TXT	[TEX,DEK]	1	4480
1979-02-22 21:52	TEXDIF.PAR	[TEX,DEK]	1	33920
1979-03-31 19:32	TEXDOC.SAI	[TEX,DEK]	1	51200
1979-04-07 08:45	TEXDOC.SAI	[TEX,DEK]	2	51200
1979-04-07 08:45	TEXDOC.SAI	[TEX,DEK]	3	51200
1979-06-07 22:22	TEXDOC.SAI	[TEX,DEK]	4	51200
1979-10-25 16:51	TEXDVI.SAI	[TEX,DEK]	1	16000
1980-03-23 15:24	TEXDVI.SAI	[TEX,DEK]	2	16000
1981-01-05 15:47	TEXDVI.SAI	[TEX,DEK]	3	16000
1981-02-27 15:27	TEXDVI.SAI	[TEX,DEK]	4	17920
1981-05-03 15:11	TEXDVI.SAI	[TEX,DEK]	5	17920
1981-07-17 21:22	TEXDVI.SAI	[TEX,DEK]	6	17920
1982-04-20 21:34	TEXDVI.SAI	[TEX,DEK]	7	18560
1980-03-28 21:49	TEXEXT.DIF	[TEX,DEK]	1	4110
1978-08-27 13:13	TEXEXT.OLD	[TEX,DEK]	1	2560
1978-03-10 13:56	TEXEXT.SAI	[TEX,DEK]	1	2560
1980-01-25 08:01	TEXEXT.SAI	[TEX,DEK]	2	3345
1980-04-23 18:45	TEXEXT.SAI	[TEX,DEK]	3	11520
1980-06-16 12:27	TEXEXT.SAI	[TEX,DEK]	4	12160
1981-01-05 14:33	TEXEXT.SAI	[TEX,DEK]	5	12160
1981-03-05 07:57	TEXEXT.SAI	[TEX,DEK]	6	12160
1979-10-25 16:40	TEXF20.SAI	[TEX,DEK]	1	3995
1980-03-25 23:26	TEXF20.SAI	[TEX,DEK]	2	4480
1981-03-06 10:55	TEXF20.SAI	[TEX,DEK]	3	4480
1980-03-28 21:48	TEXFIL.DIF	[TEX,DEK]	1	1535
1979-10-15 09:14	TEXFIL.SAI	[TEX,DEK]	1	4480
1980-01-25 08:02	TEXFIL.SAI	[TEX,DEK]	2	4615
1980-03-28 21:43	TEXFIL.SAI	[TEX,DEK]	3	4530
1981-01-05 13:27	TEXFIL.SAI	[TEX,DEK]	4	3320
1981-02-26 09:18	TEXFIL.SAI	[TEX,DEK]	5	3840
1978-08-01 18:16	TEXFNT.TEX	[TEX,DEK]	1	3840
1979-01-21 11:09	TEXFNT.TEX	[TEX,DEK]	2	3840
1980-03-26 17:15	TEXFNT.TEX	[TEX,DEK]	3	3840
1980-03-28 21:47	TEXHDR.DIF	[TEX,DEK]	1	1305
1978-08-27 13:13	TEXHDR.OLD	[TEX,DEK]	1	31360
1978-06-10 11:58	TEXHDR.SAI	[TEX,DEK]	1	30080
1978-06-21 04:25	TEXHDR.SAI	[TEX,DEK]	2	30080
1978-08-07 20:31	TEXHDR.SAI	[TEX,DEK]	3	31360
1978-08-28 12:49	TEXHDR.SAI	[TEX,DEK]	4	31360
1978-09-08 04:02	TEXHDR.SAI	[TEX,DEK]	5	31360
1978-12-06 08:36	TEXHDR.SAI	[TEX,DEK]	6	31360
1978-12-11 10:59	TEXHDR.SAI	[TEX,DEK]	7	31360
1979-01-27 11:08	TEXHDR.SAI	[TEX,DEK]	8	31360

perm filename TEXDVI.SAI[TEX,DEK] blob sn#653724 filedate 1982-04-20 generic text, type C, neo UTF8

```
COMMENT ⊗   VALID 00008 PAGES
C REC  PAGE   DESCRIPTION
C00001 00001
C00002 00002    entry begin comment The output module of TEX.
C00004 00003    initialization: initout,declareofil
C00008 00004    General description of the shipout procedure.
C00010 00005    some macro and subroutine definitions
C00014 00006    The recursive traversal procedures: vlistout,hlistout
C00026 00007    internal procedure shipout(integer p) # the main output proced
C00028 00008    internal procedure closeout # just before TEX stops, do this
C00030 ENDMK
C⊗;
entry; begin comment The output module of TEX.

(It is wise to read the box data structure definitions in TEXSEM
before going very deeply into the following code.)

Each TEXOUT module is supposed to include the following procedures
invoked by the main program:

        initout                 gets the output module started initially
        declareofil(string s)   called when the output file name is known
        shipout(integer p)      called for each nonempty page to be output
        closeout                finishes the output

This routine produces output in "device independent" format, described below.
This format is contains enough information to probably easily be transformed
into a format suitable for a wide range of actual devices.  (It is certainly
sufficient for our Versatec and Alphatype).  This program has been hacked by
DRF from the original XGP output routine written by DEK.

Routines for other devices will probably have a rather similar structure;

require "TEXHDR.SAI" source_file;
require "
        Note: This output module prepares device-independent files. " message;
require "{}{}" delimiters; "used for macros" comment for SCORE ;
comment initialization: initout,declareofil;

IFTENEX external integer _skip_; ENDTENEX
internal string ofilext # filename extension for output;
internal string deviceext # extension to use in font information files;
internal string ofilname # output file name, set by first \input;
internal string libraryarea # default system area for fonts, also look here
                              for \input things;
saf integer array fontused[0:nfonts-1] # mark if font has been used;
integer dvichan # output channel number;

boolean nooutputyet # is there at least one page in the DVI file? ;
integer lastfont # most recently used font;
real maxpagewidth,maxpageheight;
integer thispageptr,lastpageptr;

# these things are PUSHed and POPed ;
real x,y # what the dvi file thinks they are;
integer wamt,xamt,yamt,zamt # ditto;
```


7840
7840
7840
7840
7840
640
640
1280
1280
1280
1280
5600
0320
5280
5280
4000
0480
6240
6240
6240
6240
3920
@d coNtribute=80 {go here to link a node into the
@p procedure build_page; {append contributions to
label exit,done,done1(contribute,update_heights;
var p:pointer; {the node being appended}
@!q,@!r,@!s:@A←S]i∃dvAw9←IKf↓EKS]≤AKqC5S]KIx~∃CAe
KIKgM←dA←_Ayayx~∃Cλu$]i∃OKdv↓wECI9KgfA=HAGkIeK]h↓
CYKHlAwgSiKfAkMKHAM=dAS]MKeiS=\AGC1GkYCQS←]gx~∃C]C
keeK9hAaC≥JvAS_AShA%fAiS5JAM←HABAa¬OJAEIKCVX4∀UakF
←]ie%D1QK¬HR{]UYXv~)y≠C-JAiQ∀AG←]QeSEkQS←\A1SghA
←]ie%D1QK¬Iy|1~∃Kq%huK]⊂v~∀~)Ay5CWJAQQJAG=]ieS ki
Kf\AM←[JAAK←aY∀AGCY0AiQSLAk]gQekGiUeKH~)ae←OIC[[
keeK9hAaC≥Jv@\8]|z4∃yU_AiQJ↓Gkee∃]hAa¬OJASLAK[aQrA
keeK9hAaC≥Jv~∀%SLAi!SfA]=IJASLAC\A%]gKeQS←\X↓yO←i
SCiK⊂AoSi AiQSLAEeK¬Wa←S9i|v4∃yπ!KGVA%LA]←⊃JAyapA
sGYJ↓]←IJ↓yayxv~∃I=]Jt~(~∃Ay→S],A]←I∀Ayax↓S]i↑↓iQ
keeK9hAaC≥JAC]⊂AyO←Q↑AI←9Ky|t~∃YS9VQaC≥J1iC%XS?`l
YJA]=IJAyAy|z4∃YS],QG←]QeSD1!KCHS⁄YS]V!`RvA1S]VQ@
←]ie%EkiKq|v~)oQCiMSh1]=IJtAy!eKACeJAQ↑A[←YJAoQ¬i

Aufspaltung von Software in Entitäten



Module

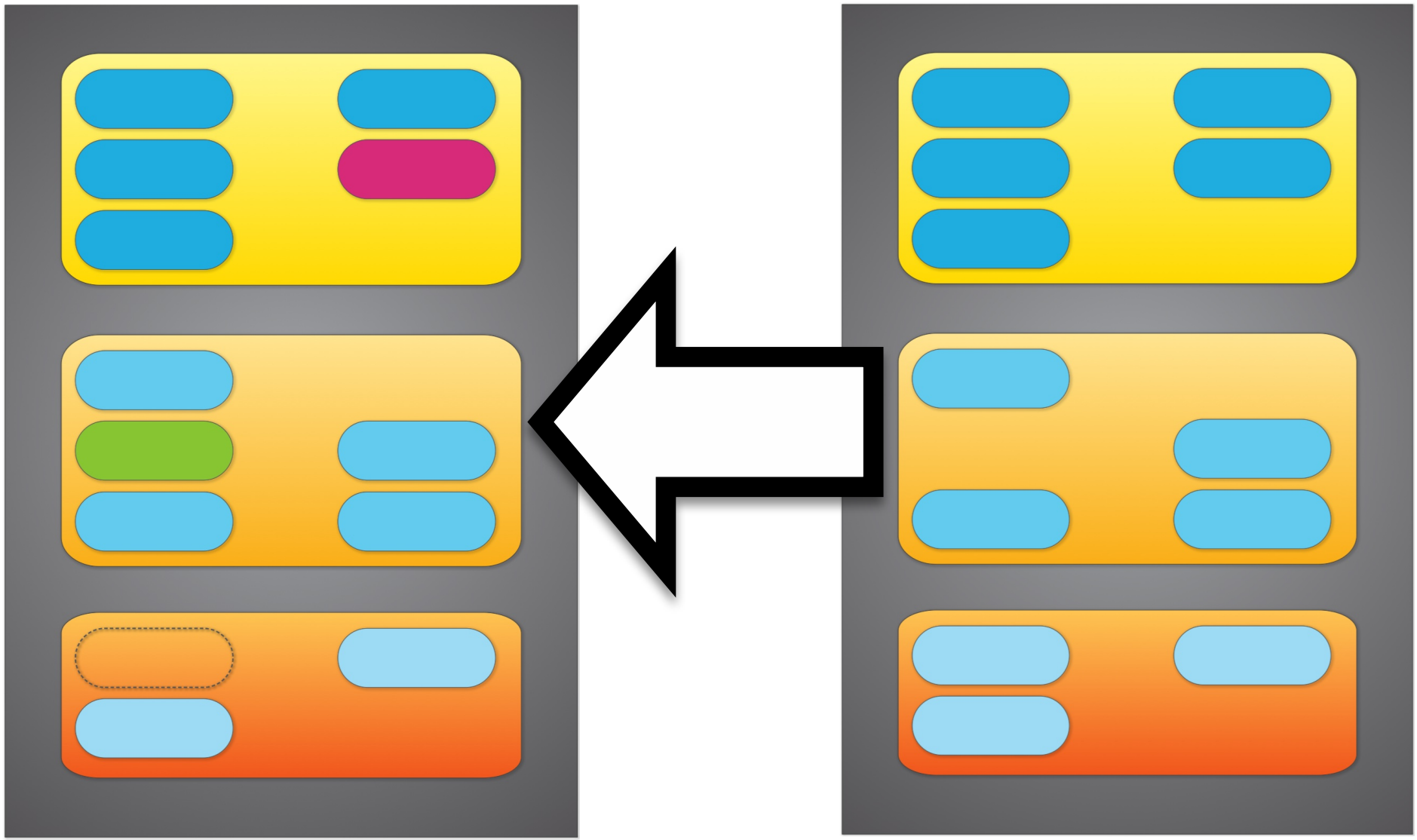
Dateien

Klassen

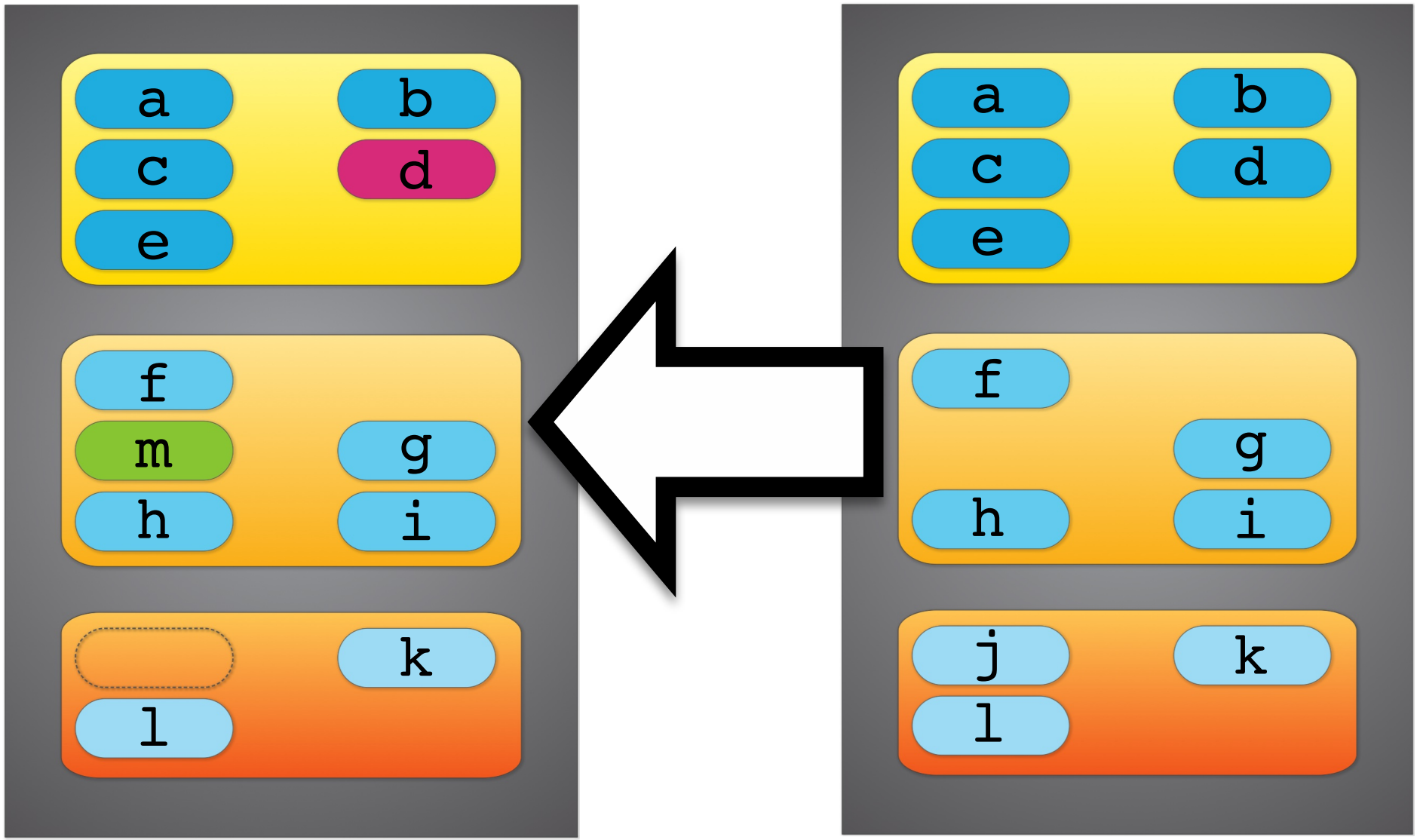
Routinen

Variablen

Origin-Analyse



Origin-Analyse



Knuths Fehlerprotokollierung

19 May 1978

249 Add a `\topbaseline` feature [later called `\topskip`]. §1001 G

245 ↦ **250** Subtract the math spacing change of May 14. §760 Q

251 Skip past blanks in the *scan_math* procedure. [This blank-skipping will eventually go into *scan_left_brace*.] §403 A

252 Introduce a *missing_brace* routine [later generalized] to improve error recovery in *mmode* + *math_shift*, when the top of *save_stack* isn't a *math_shift_group*. §1065 I

253 Adjust the math spacing between closing parentheses and Ord, Op, Open, Punct. §764 Q

254 Make the underline go further under. §735 Q

96 ↦ **255** Compute the proper natural width when a displayed equation follows a paragraph whose fillglue has been deleted by *line_break*. §1146 S

20 May 1978

256 Fix the spurious value of *prev_depth* inside alignments. §775 A

257 Consider (and defeat) the following scenario: The u and v lists are built in *init_align* using *temp_head*; then while scanning '`\tabskip 2pt\rt{...}`' the macro `\rt` is expanded, clobbering *temp_head*. §779 S

- *That bug was more subtle than usual.*

258 Add the parameter *num3*, so that the positioning of `\atop` can be different from that for fractions. §700 Q

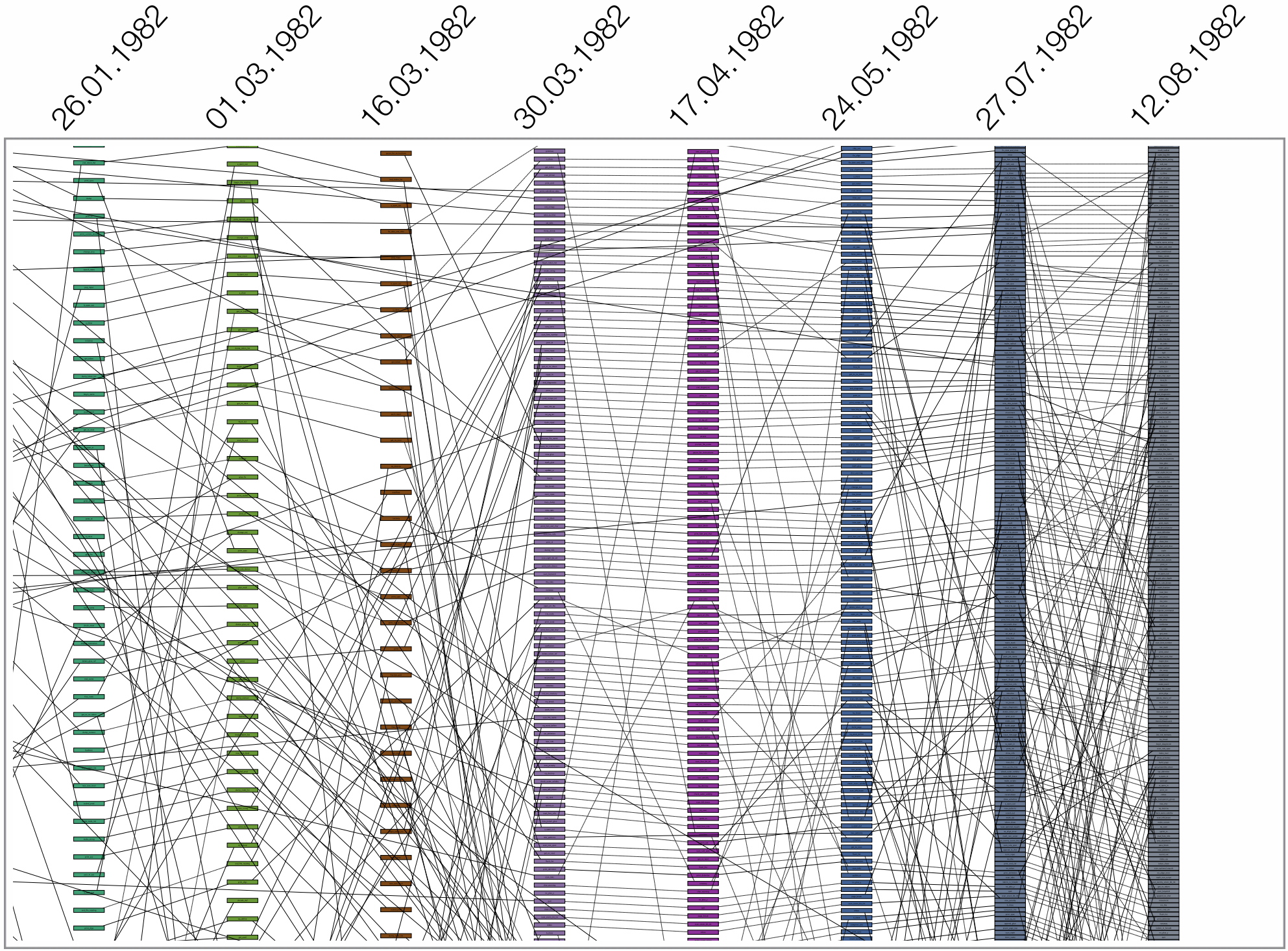
259 Add new parameters *delim1* and *delim2*, so that `\comb` can use fixed size delimiters, not computed as with `\left`. §748 Q

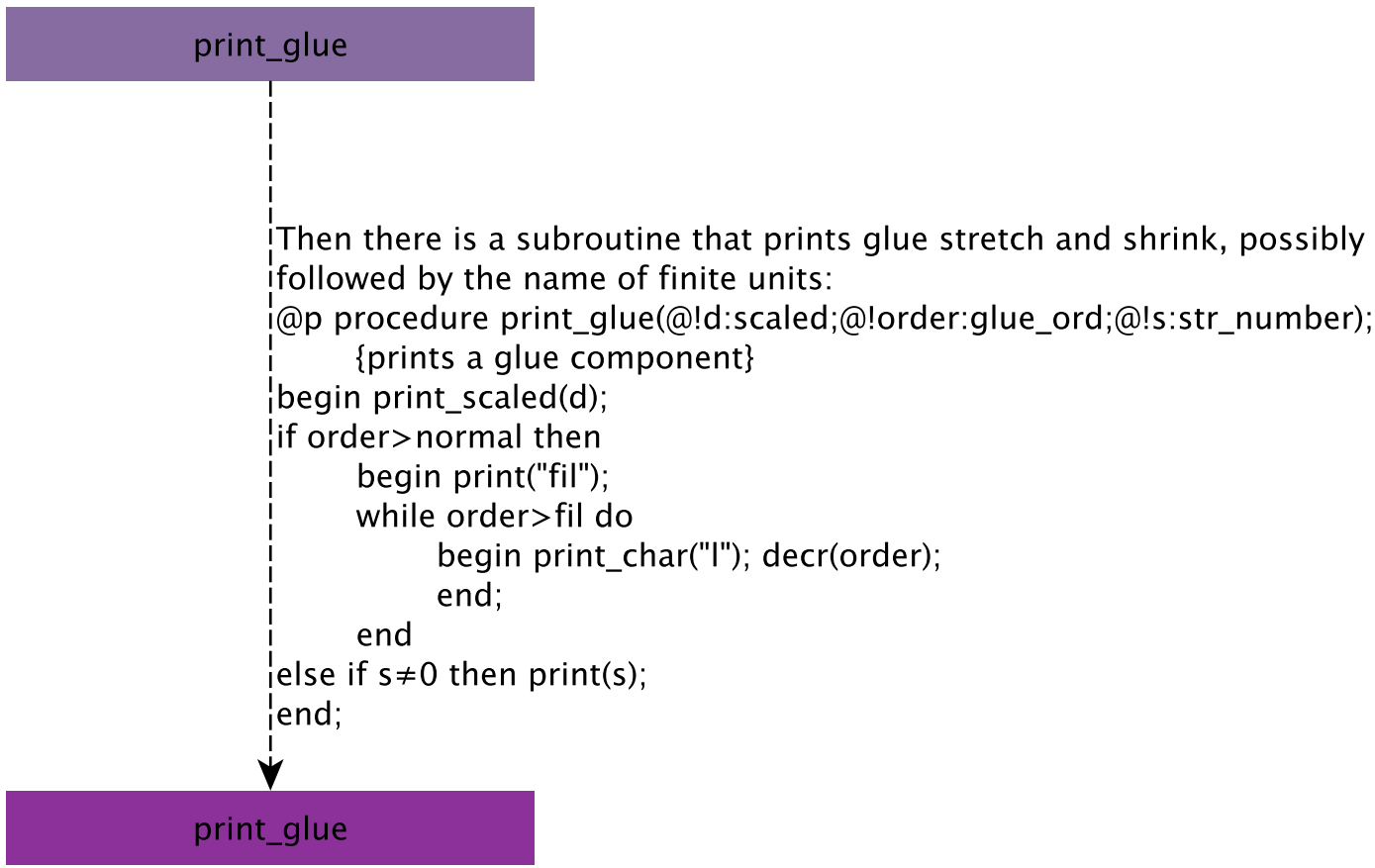
Fehlerklassen

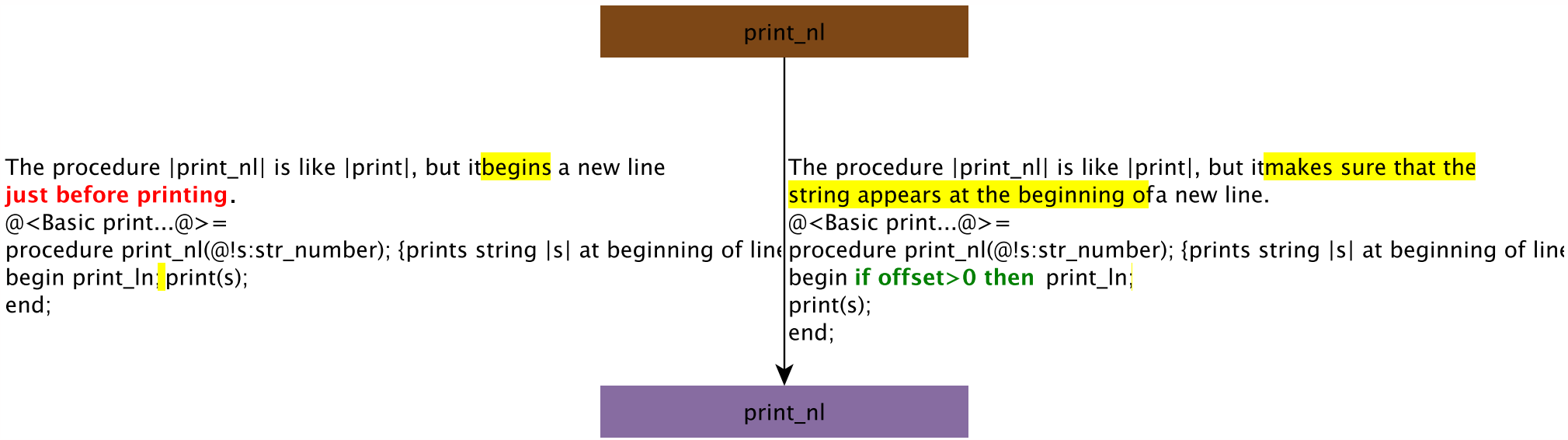
A	algorithm awry	⚠
B	blunder	⚠
C	consistency / clarity cleanup	⬆
D	data structure debacle	⚠
E	efficiency enhancement	⬆
F	forgotten function	⚠
G	generalization / growth of ability	⬆
I	interactive improvement	⬆
L	language liability	⚠
M	mismatch between modules	⚠
P	promotion of portability	⬆
Q	quest for quality	⬆
R	reinforcement of robustness	⚠
S	surprising scenario	⚠
T	trivial typo	⚠

„Forgot to include [...] in *showmem*“
„Include [...] in *check_mem*“

„[...] in loop of *dumplist* [...]“
„[...] into the loop of *show_token_list* [...]“


26.01.1982
01.03.1982
16.03.1982
30.03.1982
17.04.1982
24.05.1982
27.07.1982
12.08.1982


print_glue
Then there is a subroutine that prints glue stretch and shrink, possibly
followed by the name of finite units:
@p procedure print_glue(@!d:scaled;@!order:glue_ord;@!s:str_number);
{prints a glue component}
begin print_scaled(d);
if order>normal then
begin print("fil");
while order>fil do
begin print_char("l"); decr(order);
end;
end
else if s≠0 then print(s);
end;
print_glue


print_nl
The procedure |print_nl| is like |print|, but itbegins a new line
just before printing.
@<Basic print...@>=
procedure print_nl(@!s:str_number); {prints string |s| at beginning of line
begin print_ln; print(s);
end;
The procedure |print_nl| is like |print|, but itmakes sure that the
string appears at the beginning ofa new line.
@<Basic print...@>=
procedure print_nl(@!s:str_number); {prints string |s| at beginning of line
begin if offset>0 then print_ln;
print(s);
end;
print_nl

```
        nodesize(p)←s # reset the node size in case it grew;
        p←link(p);
        end until p=rover # repeat until whole list traversed;
overflow(varsize) # no large enough space was found;
end;
```

getnode

```
internal integer procedure getnode(integer size);
begin comment returns a pointer to a new node of the specified size,
which must be 2 or more;
integer p,q,s,t,u;
comment The following tricky code does
        llink(rlink(p))←llink(p), rlink(llink(p))←rlink(p);
define removenode(p)=
        ⊂begin if p=rover then
                begin rover←link(p);
                if p=rover then  overflow(varsize) # list musn't become empty;
                end;
        u←((p lsh infod) + p) xor mem[p] # bits to change;
        t←field(llink,mem[p]) # llink(p);
        mem[t]←field(link,u) xor mem[t];
        t←link(p) # rlink(p);
        mem[t]←ufield(info,u) xor mem[t];
        end⊃;
p←rover;
do      begin q←p+nodesize(p) # q points past the end of node(p);
        while mem[q]                begin comment merge with the next node, if it is free too;
                removenode(q); q←q+nodesize(q);
                end;
        if (s←q-p) ≥ size+2 then
                begin q←q-size # allocate from top end;
                nodesize(p)←q-p # remaining free area size;
                rover ← p # let rover rove around;
                 mem[q]←0; return(q) # deliver the goods;
                end;
        if s = size then
                begin removenode(p) # exact fit, now t = rlink(p);
                rover ← t # let rover rove;
                 mem[p]←0; return(p) # deliver the goods;
                end;
        nodesize(p)←s # reset the node size in case it grew;
        p←link(p);
        end until p=rover # repeat until whole list traversed;
overflow(varsize) # no large enough space was found;
end;
```

```
internal integer procedure getnode(integer size);
begin comment returns a pointer to a new node of the specified size,
which must be 2 or more. All words of the new node are set to zero;
integer p,q,s,t,u;
label ovfl, found;
comment The following tricky code does
        llink(rlink(p))←llink(p), rlink(llink(p))←rlink(p);
define removenode(p)=
        ⊂begin if p=rover then
                begin rover←link(p);
                if p=rover then  go to ovfl # list musn't become empty;
                end;
        u←((p lsh infod) + p) xor mem[p] # bits to change;
        t←field(llink,mem[p]) # llink(p);
        mem[t]←field(link,u) xor mem[t];
        t←link(p) # rlink(p);
        mem[t]←ufield(info,u) xor mem[t];
        end⊃;
p←rover;
do      begin q←p+nodesize(p) # q points past the end of node(p);
        while mem[q]                begin comment merge with the next node, if it is free too;
                removenode(q); q←q+nodesize(q);
                end;
        if (s←q-p) ≥ size+2 then
                begin q←q-size # allocate from top end;
                nodesize(p)←q-p # remaining free area size;
                rover ← p # let rover rove around;
                 go to found;
                end;
        if s = size then
                begin removenode(p) # exact fit, now t = rlink(p);
                rover ← t # let rover rove;
                 q ← p; go to found;
                end;
        nodesize(p)←s # reset the node size in case it grew;
        p←link(p);
        end until p=rover # repeat until whole list traversed;
found: for p ← q thru q+size-1 do mem[p]←0 # clear out the node found;
return(q) # deliver the goods;
ovfl: overflow(varsize) # no large enough space was found;
end;
```

getnode

```
internal integer procedure getnode(integer size);
```

```
internal integer procedure getnode(integer size)# variable-size node allocation;
```

Vielen Dank !