

Das accessibility-Paket

Babett Schalitz

Version 2.0.3, 2. November 2019

Inhaltsverzeichnis

1	Einleitung	4
1.1	Einige Warnungen	4
1.2	Urheberrechtshinweise	4
2	Benutzerschnittstelle	5
2.1	Wie man das Paket einbindet	5
2.2	Optionen	5
2.3	Die Befehle	6
3	Die Implementierung	8
3.1	Der Vorspann	8
3.1.1	Paketinformationen und benötigte Pakete	8
3.1.2	Variablendeklaration	8
3.1.3	Definition der Optionen	9
3.1.4	Überprüfen des Ausgabemodus	10
3.1.5	Überprüfen der Dokumentenklasse	10
3.1.6	Definition der neuen Befehle	11
3.2	allgemeine Hilfsmakros	11
3.2.1	Der Stack	11
3.2.2	Reine Strukturelemente	13
3.2.3	Normale und besondere Textelemente	15
3.2.4	Elemente auf Zeilenebene	18
3.2.5	Marked Content	19
3.3	Erkennen von Absätzen	22
3.4	Dokumentbeginn	23
3.5	Dokumentende	23
3.6	Seitenumbruch	25
3.6.1	Automatischer Seitenumbruch	25
3.6.2	Manueller Seitenumbruch	32
3.7	Überschriften	32
3.7.1	Hilfsmakro	32
3.7.2	Kapitel	33
3.7.3	Überschriften mit Afterskip	34
3.7.4	Überschriften ohne Afterskip	36
3.7.5	Minisec	37
3.8	Blockelemente	38
3.8.1	Zitatumgebungen	38
3.8.2	Verbatim, Listings und andere	39
3.8.3	Theorem	40
3.8.4	Aufzählumgebungen	41
3.8.5	Formeln	45
3.8.6	Gleitumgebungen	46

3.8.7	Caption	47
3.8.8	Tabellen	48
3.9	Elemente auf Zeilenebene	51
3.9.1	Texthervorhebungen	51
3.9.2	Verweise auf andere Textstellen	52
3.9.3	eingebettete Objekte im Textfluss	54
3.9.4	Fußnoten	55
3.10	Verzeichnisse	57
3.10.1	Inhaltsverzeichnis und die Listen der Float-Objekte	57
3.10.2	Literaturverzeichnis	59
3.10.3	Index	60
3.11	Layoutbefehle	61
3.11.1	Kopf- und Fußzeilen als Artefakte	61
3.11.2	Linien als Artefakte	62
3.11.3	Titelseite	64
3.12	Verträglichkeit mit anderen Dokumentklassen	65
3.13	Verträglichkeit mit anderen Paketen	65
3.13.1	Das multicolumn-Paket	65
3.13.2	Das graphics-Paket	65
3.13.3	Das picture-Paket	66
3.13.4	Das babel-Paket	66
3.13.5	Das makeidx-Paket	69
3.13.6	Das glossary-Paket	69
3.13.7	Das booktabs-Paket	72
3.13.8	Das hyperref-Paket	72
3.13.9	Das caption-Paket	72
3.13.10	Das tabularx-Paket	72
3.13.11	Das longtabular-Paket	72
3.13.12	Das color-Paket	72
3.13.13	Das theorem-Paket	72
3.13.14	Das thmbox-Paket	73
3.13.15	Das listings-Paket	73
3.13.16	Das scrpage2-Paket	73

Literaturverzeichnis

74

1 Einleitung

Das accessibility-Paket bietet die Möglichkeit „Tagged PDF“ zu erstellen, das heißt vorhandene \LaTeX -Strukturen können in das fertige PDF übernommen werden, was insbesondere die Accessibility des erzeugten PDF steigert.

Es ermöglicht eine bessere Weiterverwendung von Textinhalten, zudem können etliche Funktionen besser automatisiert werden.

- Z. B. können Screenreader dem Anwender das Dokument unter Nutzung der Strukturen vorlesen. Zum einen ist eine Unterscheidung zwischen Überschriften und Haupttext für ihn überhaupt erst möglich. Die visuellen Hervorhebungen wie Schriftart, -größe oder Farbe waren für blinde Anwender nicht wahrnehmbar. Zum anderen wird die Erstellung von z. B. Überschriftenlisten realisierbar, mit deren Hilfe der Nutzer mit Sehbeeinträchtigung im Dokument besser navigieren kann, indem er eine interessante Überschrift direkt anspringt.
- Prinzipiell können Tagged PDF automatisch „Umfließen“, sich also ähnlich wie XHTML-Dokumente im Browser an die jeweils verfügbare Darstellungsfläche anpassen. Dieses Feature wird durch eine Besonderheit in pdftex im Moment nicht unterstützt (vgl. [Sch07b]).
- Die weitere Konvertierung des PDF-Dokumentes in andere Formate wird zuverlässiger. Bei „Speichern unter...“ gehen momentan sämtliche Leerzeichen verloren, das resultiert gleichermaßen aus dem eben genannten Problem.

1.1 Einige Warnungen

Die Struktur kann mit dem gewählten Vorgehen nur in PDF-Dokumenten erhalten werden, die mit pdftex direkt erzeugt werden. Transformationen über das DVI- oder PS-Format in PDF werden nicht unterstützt.

Bisher ist leider eine zuverlässige Erkennung von Seitenumbrüchen nicht möglich. Des Weiteren wurde dieses Paket unter Verwendung der Dokumentenklasse |scrrept| entwickelt und arbeitet damit am zuverlässigsten. Ein Test mit anderen Klassen des Koma-Script-Paketes und den Standardklassen ist teilweise erfolgt. Mehr Aufwand konnte im Rahmen der Diplomarbeit leider nicht betrieben werden.

1.2 Urheberrechtshinweise

Dieses Programm kann weitergegeben und/oder verändert werden unter den Bedingungen des \LaTeX Projekt Public License die unter CTAN (im Verzeichnis macros/latex/base/lppl.txt) archiviert ist. An Weiterentwicklung oder Verbesserungsvorschlägen ist die Autorin sehr interessiert. Auch Fragen, Kritik oder sonstige Anregungen können an [Github](#) gerichtet werden.

2 Benutzerschnittstelle

2.1 Wie man das Paket einbindet

Grundsätzlich wird das Paket einfach in der Dokumentenpräambel geladen. Es sollte allerdings möglichst nach allen andere Paketen geladen werden, insbesondere nach hyperref.

```
\documentclass{scrrept}
\usepackage[Optionen]{accessibility}
\begin{document}
...
\end{document}
```

Die verfügbaren Optionen werden im nächsten Abschnitt vorgestellt.

Sollten Sie bislang nicht mit pdftex gearbeitet haben, ist zu beachten, dass zur korrekten Auflösung sämtlicher Referenzen teilweise mehrere Durchläufe notwendig sind. Der Aufruf auf der Kommandozeile erfolgt analog zur Verarbeitung mittels latex.

```
pdflatex dateiname
    Aufrufe von BibTex, MakeIndex
pdflatex dateiname
pdflatex dateiname
```

Nach dem ersten Durchlauf, ist der Quelltext der PDF-Datei teilweise nicht korrekt, das heißt bestimmte Teile stehen doppelt drin, so dass zu Darstellungsproblemen im Adobe Reader kommen kann.

2.2 Optionen

Eine Liste der verfügbaren Optionen und eine kurze Erläuterung zeigt die nachfolgende Tabelle 2.1.

Option	Beschreibung
untagged	Keine Strukturinformationen
tagged	PDF mit Strukturinformationen
flatstructure	Erzeugt eine flache Struktur.
highstructure	Erzeugt eine verschachtelte Struktur.

Tabelle 2.1: Überblick über alle verfügbaren Optionen

Dabei kann entweder eine verschachtelte oder eine flache Struktur erzeugt werden. Ebenso verhält es sich mit den Optionen untagged und tagged. Gibt man keine Optionen an, so wird

ein PDF mit den Standardoptionen erzeugt. D. h. es wird Tagged PDF mit einer geschachtelten Struktur erzeugt.

Bei der flachen Struktur werden alle weiteren Elemente direkt unter dem Wurzelement in den Baum eingefügt. Es entsteht eine mit XHTML vergleichbare Struktur (vgl. Abbildung 2.1).

Die Latex-Struktur

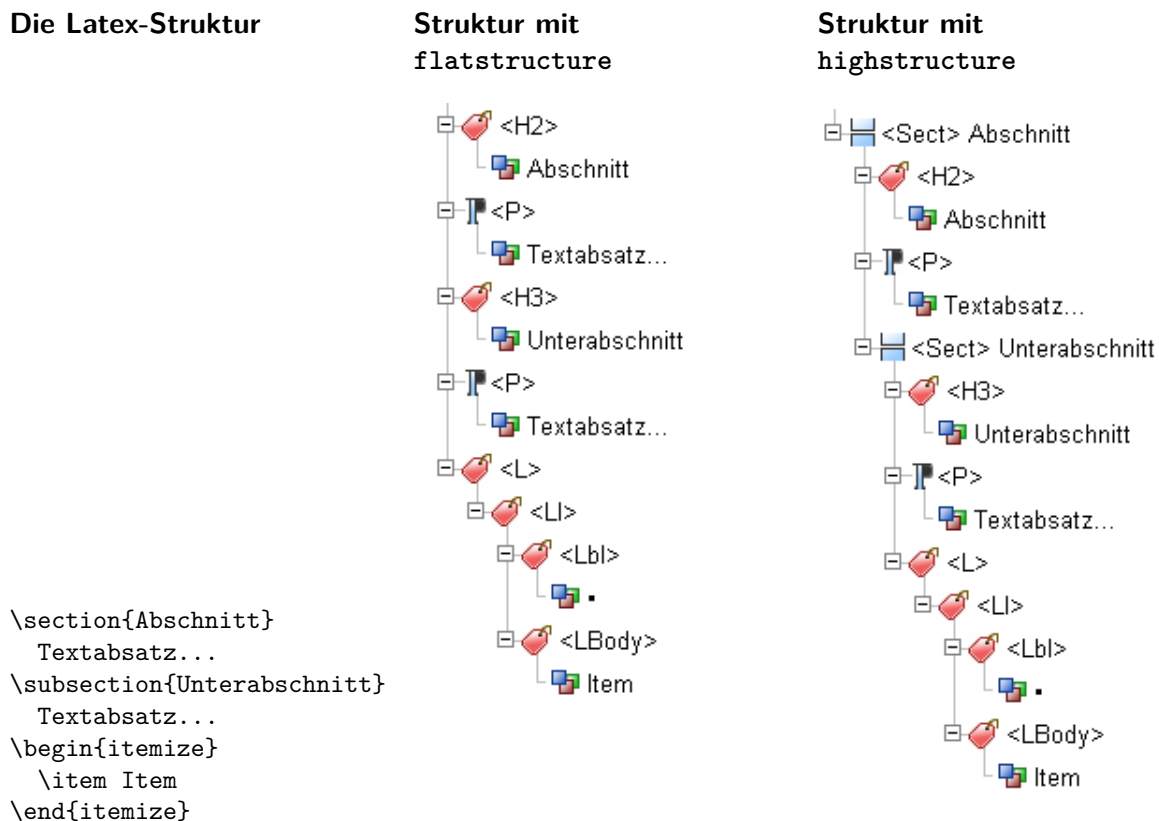


Abbildung 2.1: Erläuterungen zu flachen und strukturierten Variante

Unter Verwendung der Option `highstructure` wird eine durch `/Sect`-Elemente tiefer verschachtelte Struktur erzeugt. Gerade in größeren, gut strukturierten Latex-Dokumenten enthält der Baum auf der ersten Ebene nur die `/Sect`-Objekte der Kapitel oder Teile (Parts), je nachdem welche die höchste Ebene der Dokumentenklasse ist. Für längere Dokumente ist diese Variante übersichtlicher. Für kürzere Dokumente hingegen ist die flache Strukturierung durchaus ausreichend.

2.3 Die Befehle

Für den normalen Autor führt das Paket `accessibility` nur wenige neue Befehle ein. Es erzeugt die Struktur vielmehr durch bestmögliches transparentes Umdefinieren der Standard-Latex-Befehle. Diese können größtenteils wie gewohnt verwendet werden. Eine ausführliche Anleitung finden Sie in der zugehörigen Autorenanleitung [Sch07b].

Neue Befehle dienen der Erhöhung der Accessibility im Ergebnisdokument, also dem PDF. Für Grafiken und Formeln steht nun ein Befehl `\alt` für alternative Beschreibungen bereit. Er muss nach Möglichkeit am Anfang der Umgebung stehen und sollte reinen ASCII-Text enthalten. Die Zeichen „`^`“, `{`, `}`, `[`, `]`, `_`“ können verwendet werden, auf die Verwendung des „`\`“ ist hingegen zu verzichten. Eine mögliche Verwendung zeigt die Abbildung 2.2.

Einbinden einer Grafik

```
\begin{figure}[htbp]
  \alt{Hier die alternative
  Beschreibung der Figure angeben.}
  \includegraphics{beispielbild}
  \caption{Beispielbild}
\end{figure}
```

Einbindung einer Formel

```
\begin{equation}
  \alt{c = \sqrt{a^2+b^2}}
  c = \sqrt{ a^2+b^2 }
\end{equation}
```

Abbildung 2.2: Beispiel für die Verwendung alternativen Beschreibungen

Des Weiteren ist insbesondere bei der Beschreibung von Formeln von der Wiedergaben von Layoutbefehlen (wie fett, kursiv oder Ausrichtungsbefehle) abzuraten. Es sollte auf eine sinnvolle Strukturierung der Beschreibung mittels Leerzeichen und eindeutige Klammerung geachtet werden.

3 Die Implementierung

Die Implementierung basiert auf der Manipulation des PDF-Outputs über die Schnittstelle von `pdftex`. Dabei werden insbesondere die Befehle `\pdfliteral` und `\pdfobj` genutzt. Diese Primitiven fügen den übergebenen Text direkt in den Quellcode der PDF-Datei ein. Er muss der zugrunde liegenden Spezifikation folglich entsprechen. Ansonsten wird ein nicht valides Dokument erzeugt.

Für detailliertere Ausführungen, wie und warum das Paket `accessibility` entstand, ist die Diplomarbeit „Erhöhung von Accessibility in \LaTeX -Dokumenten“ [Sch07a] zu konsultieren. Sie enthält ein umfassendes Konzept sowie tiefer gehende Erläuterungen zum PDF.

3.1 Der Vorspann

3.1.1 Paketinformationen und benötigte Pakete

Dieses Paket sollte mit allen \LaTeX 2 ϵ Versionen zusammenarbeiten, wurde aber nur mit der Version vom 1. Juni 2000 getestet.

```
1 {*package}
2 \ProvidesPackage{accessibility}[2019/11/02 v. 2.0.3]
3 \NeedsTeXFormat{LaTeX2e}
```

Zunächst werden einige benötigte Pakete geladen.

```
4 \RequirePackage{xkeyval}
5 \RequirePackage{ifthen}
```

3.1.2 Variablendeklaration

Die Variablen werden benötigt, um später den Strukturbaum aufzubauen. Für die Objektnummern der PDF-Objekte wird jeweils ein Zähler gebraucht.

Das Wurzelement (`/StructTreeRoot`) wird in Zähler `StructTree` gehalten. Dazu wird ein neues PDF-Objekt reserviert und die Nummer zur späteren Verwendung gespeichert. Das `Karray` dient der Speicherung sämtlicher Objektreferenzen, die dem Wurzelobjekt untergeordnet werden. Es ist anfangs leer.

```
6 \newcounter{StructTree}%
7 \pdfobj reserveobjnum%
8 \setcounter{StructTree}{\pdflastobj}%
9 \xdef\Karray{}
```

Zur kurzzeitigen Zwischenspeicherung von Objektnummern steht der Zähler `ObjHelp` zur Verfügung.

```
10 \newcounter{ObjHelp}%
```

Der Zähler TaggedObj hält die aktuelle /MCID des ausgezeichneten Objektes, um die Verbindung zum Strukturbaum herzustellen. Laut PDF-Referenz wird diese ID für jedes Seitenobjekt zurückgesetzt. Da der Seitenzähler aber erst nach \shipout berichtigt wird, stimmt die Seitenreferenz für die bis dahin geschriebenen Objekte nicht. Es kommt zu doppelten ID auf einer Seite, was die eindeutige Zuordnung stört und zahlreiche Fehler birgt. Folgefehler dieses Problems können durch die durchgehenden Nummerierung beseitigt werden.

```
11 \newcounter{TaggedObj}%[page]
```

In dem Schalter ACCESSProblems wird gespeichert, ob noch Bedenken bezüglich der Accessibility des Dokumentes bestehen, also z. B. alternative Texte nicht gesetzt wurden oder ähnliches.

```
12 \newboolean{ACCESSProblems} \setboolean{ACCESSProblems}{false}%
```

Diese Variablen dienen der Speicherung der aktuellen Sprache sowie der Unterscheidung, ob die Sprache geändert wurde.

```
13 \gdef\DocumentLanguage{}%
14 \gdef\ActualLanguage{}%
15 \newif\ifLanguageDiff \global\LanguageDifffalse%
16 \gdef\LanguageCode{}%
```

DetailedStructure dient der Feststellung, ob eine geschachtelte oder flache Struktur erzeugt werden soll. Während @Access@pdf wahr ist, wenn Tagged PDF erzeugt werden soll und eine geeignete pdftex-Version aktiv ist.

```
17 \newboolean{@tagged@pdf} \setboolean{@tagged@pdf}{false}%
18 \newboolean{@right@pdfversion} \setboolean{@tagged@pdf}{false}%
19 \newboolean{@Access@pdf} \setboolean{@Access@pdf}{false}%
20 \newif\ifPDFDetailedStructure \global\PDFDetailedStructuretrue%
```

3.1.3 Definition der Optionen

Hier werden die möglichen Optionen deklariert und passende Variablen für die Weiternutzung initialisiert.

```
21 \DeclareOption{flatstructure}{\global\PDFDetailedStructurefalse}%
22 \DeclareOption{highstructure}{\global\PDFDetailedStructuretrue}%
23 \DeclareOption{tagged}{\setboolean{@tagged@pdf}{true}}%
24 \DeclareOption{untagged}{\setboolean{@tagged@pdf}{false}}%
25 \DeclareOption*{%
26   \PackageWarning{accessibility}{Unknown Option \CurrentOption}}%
27 \ProcessOptions\relax%
```

3.1.4 Überprüfen des Ausgabemodus

An dieser Stelle wird der Ausgabemodus sowie die verwandte PDFTEX-Version getestet, erst ab der Version 1.20 kann direkter PDF-Output generiert werden.

```
28 \ifthenelse{\isundefined{\pdfoutput}}{%
29   %latex with dvips
30   \setboolean{@right@pdfversion}{false}%
31 }{\ifthenelse{\number\pdfoutput<1}{%
32   %pdflatex in DVI mode
33   \setboolean{@right@pdfversion}{false}%
34 }{\pdflatex in PDF mode
35   \ifthenelse{\pdftexversion<120}{%
36     \PackageError{accessibility}%
37     {pdfTeX/pdfLaTeX version >= 1.20 required for direct PDF outut}%
38     {Try to install a more recent version!}%
39   }{%
40     %It is the right version
41     \setboolean{@right@pdfversion}{true}%
42   }%
43 }%
44 }
```

Nur wenn beide Bedingungen erfüllt sind, wird im weiteren Verlauf „Tagged“ PDF erzeugt.

```
45 \ifthenelse{\boolean{@right@pdfversion} \and \boolean{@tagged@pdf}}{%
46   \setboolean{@Access@pdf}{true}%
47 }{%
48   \setboolean{@Access@pdf}{false}%
49 }
```

3.1.5 Überprüfen der Dokumentenklasse

Da die bereitgestellten logischen Befehle je nach gewählter Dokumentenklasse variieren, wird hier zwischen den Standardklassen und denen des KOMA-Scripts unterschieden.

```
50 \newboolean{@KOMAScriptClass} \setboolean{@KOMAScriptClass}{false}%
51
52 \@ifclassloaded{scrreprt} {\setboolean{@KOMAScriptClass}{true}}{%
53 \@ifclassloaded{scrbook}  {\setboolean{@KOMAScriptClass}{true}}{%
54 \@ifclassloaded{scrartcl} {\setboolean{@KOMAScriptClass}{true}}{%
55 \ifthenelse{\boolean{@KOMAScriptClass}}{%
56   \PackageInfo{accessibility}{KOMAScript Klasse}}{%
57
58 \newboolean{@StandardClass} \setboolean{@StandardClass}{false}%
59
60 \@ifclassloaded{report} {\setboolean{@StandardClass}{true}}{%
61 \@ifclassloaded{book}  {\setboolean{@StandardClass}{true}}{%
62 \@ifclassloaded{article}{\setboolean{@StandardClass}{true}}{%
63
64 \ifthenelse{\boolean{@StandardClass}}{%
```

Noch einige sinnvolle Variablenbelegungen zur PDF-Erzeugung. Sie müssen im fertigen Code nicht mehr enthalten sein.

```
66 \pdfcompresslevel=0% Damit wird die PDF-Quelldatei lesbar
67 \pdfminorversion=6% Bestimmt die PDF - Version der Ausgabe
68 %\pdfadjustspacing=0% 0, 1 oder 2 Änderung nicht erkannt
```

3.1.6 Definition der neuen Befehle

An dieser Stelle werden die neu eingeführten Befehle für die benötigten Zusatzinformationen definiert.

```
69 \newcommand{\alt}[1]{\xdef\altAttr{#1}}%
70 \newcommand{\newhref}[3]{\xdef\altAttr{#2}\href{#1}{#3}}%
71 %
72 \@ifundefined{thead}{%
73   \newcommand{\thead}[1]{%
74     \global\TableHeadCelltrue%
75     \textbf{#1}}%
76 }{%
77   \let\originalthead\thead
78   \renewcommand{\thead}{%
79     \global\TableHeadCelltrue%
80     \originalthead}%
81 }
```

3.2 allgemeine Hilfsmakros

3.2.1 Der Stack

Der Strukturbaum, lässt sich am einfachsten über einen Stack aufbauen. Prinzipiell müssen für alle Strukturelemente drei Variablen initialisiert werden, nämlich der Strukturtyp, die Objekt-Nummer und das Feld mit den Kindelementen. Für einige Elemente macht Sinn einen Titel zu generieren bzw. zu übergeben, damit wird der generische Strukturtyp näher spezifiziert.

Diese Informationen werden sowohl benötigt, um Kindelemente zu erzeugen. Als auch bei der Beendigung, also dem eigentlichen Schreiben des Strukturobjektes. Ein Zugriff ist dabei immer nur auf das oberste Element möglich. Es muss beendet werden, bevor ein darunterliegendes abgeschlossen werden kann. Für die effektive Arbeit mit dem Stack werden 3 Funktionen benötigt.

`\accessPushStack` Zum einen benötigt man eine Funktion um Elemente auf dem Stack abzu-
legen.

Parameter #1 Type #2 ObjNum #3 KidsField #4 Title

```
82 \newcount\@stackdepth \@stackdepth=0%
83 \def\accessPushStack#1#2#3#4{%
84   \ifnum \@stackdepth >15\relax%
85     \PackageWarning{accessibility}{too deep}%
86   \else%
87     \global\advance\@stackdepth\@ne%
88   \fi%
89   {\expandafter\xdef\csname StackA\romannumeral\the\@stackdepth\endcsname{#1}}%
90   {\expandafter\xdef\csname StackB\romannumeral\the\@stackdepth\endcsname{#2}}%
91   {\expandafter\xdef\csname StackC\romannumeral\the\@stackdepth\endcsname{#3}}%
92   {\expandafter\xdef\csname StackD\romannumeral\the\@stackdepth\endcsname{#4}}%
93 }%
```

`\accessPopStack` Des Weiteren ist es nötig Elemente vom Stack zu entfernen und abzarbeiten.

Parameter #1 Type #2 ObjNum #3 KidsField #4 Title

```
94 \def\accessPopStack#1#2#3#4{%
95   \ifnum \the\@stackdepth <1\relax%
96     \global\let#1\empty%
97     \global\let#2\empty%
98     \global\let#3\empty%
99     \global\let#4\empty%
100  \else%
101    \xdef#1{\csname StackA\romannumeral\the\@stackdepth\endcsname}%
102    \xdef#2{\csname StackB\romannumeral\the\@stackdepth\endcsname}%
103    \xdef#3{\csname StackC\romannumeral\the\@stackdepth\endcsname}%
104    \xdef#4{\csname StackD\romannumeral\the\@stackdepth\endcsname}%
105    %Variablen wieder leeren
106    {\expandafter\xdef\csname StackA\romannumeral\the\@stackdepth\endcsname{}}%
107    {\expandafter\xdef\csname StackB\romannumeral\the\@stackdepth\endcsname{}}%
108    {\expandafter\xdef\csname StackC\romannumeral\the\@stackdepth\endcsname{}}%
109    {\expandafter\xdef\csname StackD\romannumeral\the\@stackdepth\endcsname{}}%
110    \global\advance\@stackdepth\@m@ne%
111  \fi%
112 }%
```

`\accessReadTopStack` Zum anderen wird waren der Erzeugung von Blattknoten ein lesender Zugriff auf das oberste Stackelement benotigt. Somit kann die Objektreferenz in das Kinderfeld des Elternelementes eingetragen werden und eine Referenzierung des Elternobjektes wird moglich.

Parameter #1 Type #2 ObjNum #3 KidsField

```
113 \def\accessReadTopStack#1#2#3{%
114   \ifnum \the\@stackdepth <1\relax%
115     \global\let#1\empty%
116     \global\let#2\empty%
```

```

117     \global\let#3\empty%
118   \else%
119     \xdef#1{\csname StackA\romannumeral\the\@stackdepth\endcsname}%
120     \xdef#2{\csname StackB\romannumeral\the\@stackdepth\endcsname}%
121     \xdef#3{\csname StackC\romannumeral\the\@stackdepth\endcsname}%
122   \fi%
123 }%

```

Die folgenden Makros schreiben die tatsächlichen Elemente in die PDF-Datei, zur flexiblen Nutzung sind die übergabeparameter variabel.

3.2.2 Reine Strukturelemente

PDFStructObj Ein PDFStructObj ist eine Struktur, die dazu dient weitere Elemente zu kapseln. Die benötigten Variablen werden initialisiert und anschließend jeweils auf den Stack geschrieben.

Parameter #1 StructType #2 Title

```

124 \newenvironment{PDFStructObj}[2]{% #1 StructType #2 Title
125   \ifTextActive{\endPDFMarkContent\endPDFTextObj}\fi%
126   \pdfobj reserveobjnum% Objektnummer reservieren
127   \setcounter{ObjHelp}{\pdflastobj}%
128   \expandafter\xdef\csname PDF@#1@Array\endcsname{}%
129   \accessPushStack{#1}{\theObjHelp}{PDF@#1@Array}{#2}%drauftun
130 }%

```

Zum Abschluss eines PDFStructObj wird das oberste Element vom Stack geholt. Ist dieser leer, was sich darin zeigt, dass das StructElem leer ist, wird eine Warnung ausgegeben. Diese Abfrage erhöht die Stabilität, der Fall sollte aber normalerweise nicht auftreten.

Im jedem anderen Fall werden die Attribute geprüft und gesetzt. Anschließend wird das PDF-Objekt erzeugt und eine Referenz in das Elternelement eingefügt.

Die Erzeugung von reinen Strukturobjekten erfolgt ohne Seitenreferenz, da die Unterelemente potentiell auf mehrere Seiten verteilt sein können.

```

131 {%andere Ebenen Schließen
132   \accessPopStack\StructElem\Objnum\KidsArray\Title %runterholen
133   \ifx \StructElem\empty%
134     \PackageWarning{accessibility}{empty \string\PopStack ?}%
135   \else%
136     %\convertLanguageInCode{\languagename}%
137     \gdef\TitleHelp{}%
138     \if \Title\empty \else%
139       \gdef\TitleHelp{/T (\Title)}%
140     \fi%
141     \ifnum \@stackdepth <1\relax%
142       \xdef\ParentElem{\theStructTree}%
143       \xdef\ParentArray{Karray}%
144     \else%
145       \accessReadTopStack\ParentStructElem\ParentElem\ParentArray%

```

```

146     \fi%
147     \immediate \pdfobj useobjnum\number\Objnum{<</Type /StructElem %
148         /P \ParentElem\space 0 R %
149         \TitleHelp %
150         /C /Normal %
151         \space\LanguageCode %
152         /K [\csname \KidsArray\endcsname] %
153         /S /\StructElem>>}%
154     \pdfrefobj\Objnum%
155     \expandafter\xdef\csname \ParentArray\endcsname{%
156         \csname \ParentArray\endcsname \space \Objnum\space 0 R}%
157     \fi%
158 }

```

TODO 1 Für Blockelemente keine Sprache, tlw. zu spät geschossen → Probleme bei Sprachauszeichnung.

Implizite Beendigung von Strukturelemente

In \LaTeX werden viele Strukturen nur begonnen aber nicht explizit wieder geschlossen. Z. B. schließt eine `\section` die geöffnete `\subsection` indem sie die Zähler zurücksetzt. Es existiert folglich auch in \LaTeX eine wohl definierte Hierarchie.

`\sectionInDepth` Ordnet den Strukturelementen eine Reihenfolge zu, damit diese in der richtigen Reihenfolge automatisch geschlossen werden können.

```

159 \newcommand{\sectionInDepth}[2]{%
160     \csname #2\endcsname=100%
161     \ifthenelse{\equal{#1}{Document}}{\csname #2\endcsname=\m@ne}{}%
162     \ifthenelse{\equal{#1}{Part}}{\csname #2\endcsname=0}{}%
163     \ifthenelse{\equal{#1}{Chapter}}{\csname #2\endcsname=1}{}%
164     \ifthenelse{\equal{#1}{Section}}{\csname #2\endcsname=2}{}%
165     \ifthenelse{\equal{#1}{Subsection}}{\csname #2\endcsname=3}{}%
166     \ifthenelse{\equal{#1}{Subsubsection}}{\csname #2\endcsname=4}{}%
167     \ifthenelse{\equal{#1}{Paragraph}}{\csname #2\endcsname=5}{}%
168     \ifthenelse{\equal{#1}{Subparagraph}}{\csname #2\endcsname=6}{}%
169     \ifthenelse{\equal{#1}{Div}}{\csname #2\endcsname=7}{}%
170 }

```

`\closeUntilPDFStruct` Schließt die Sectionebenen unter Verwendung der eben definierten Reihenfolge automatisch. Damit wird die Schachtelung wesentlich flexibilisiert. Es gibt keine Fehler wenn eine Ebene fehlt.

Einige Variablen die im Macro benötigt werden.

```

171 \newcount\@bool%
172 \newcount\@elem%
173 \newcount\@elemi%

```

Zu allererst muss eventuell noch offener Text beendet werden. Anschließend beendet eine Schleife solange das jeweils oben aufliegende Stackelement, bis die nötige Tiefe erreicht ist. Ein Kapitel beendet alle Strukturen höherer Nummer, aber maximal ein Kapitel.

```

174 \newcommand{\closeUntilPDFStruct}[1]{%
175     \ifTextActive%
176         \endPDFMarkContent%
177         \endPDFTextObj%
178     \fi%
179     %Schleife
180     \@bool=0%
181     \sectionInDepth{#1}{@elemi}%
182     \ifnum \@elemi=100 \@bool=1 \fi%
183     \global\advance\@elemi\m@ne%
184     \@whilenum \@bool =0\do{%
185         \accessReadTopStack\StructElem\Objnum\KidsArray%
186         \sectionInDepth{\StructElem}{@elem}%
187         \ifthenelse{@elem >\@elemi}{%
188             \endPDFStructObj%
189         }{%
190             \@bool=1%
191         }%
192         \if #1\empty \@bool=1 \fi%
193         \ifthenelse{\equal{#1}{\StructElem}}{\@bool=1}{%
194     }%
195 }
```

3.2.3 Normale und besondere Textelemente

Sie enthalten Textabsätze und eventuell weitere Objekte auf Zeilenebene wie Fußnoten, Referenzen, Formeln, Zitat.... Ein spezielles Textelement (wie Zitat, Formel, Quellcode...) wird durch Befehle oder Umgebungen gesondert hervorgehoben. Normale Textelemente sind hingegen nicht markiert. Eine Erkennung wird durch `\everypar` erzielt. Diese Funktion wird zu Beginn jedes neuen Textabschnittes im vertikalen Modus verwendet.

Es ist immer maximal ein Textobjekt aktiv. Textobjekte können nicht ineinander geschachtelt werden.

PDFTextObj Ist eine Strukturobjekt, dass normale Textpassagen auf Absatzebene enthält.

Einige Variablen die für die folgende Definition benötigt werden.

```

196 \xdef\TextType{}%
197 \newcounter{TextObjNum}%
198 \xdef\TextArray{}%
199 \newif\ifTextActive \TextActivefalse%
200 \newif\ifSpezialTextActive \SpezialTextActivefalse%
```

Zu Beginn eines Textobjektes werden noch offene Textobjekte abgeschlossen. Anschließend werden die benötigten Variablen neu initialisiert.

```
201 \newenvironment*{PDFTextObj}{-%
202 %altes Textobj beenden, immer max. ein Textobj aktiv
203 \ifTextActive \endPDFMarkContent\endPDFTextObj\fi%
204 %neues anfangen
205 \global\TextActivetrue%
206 \pdfobj reserveobjnum% Objektnummer reservieren
207 \setcounter{TextObjNum}{\pdflastobj}%
208 \xdef\TextArray{}%
209 \xdef\TextType{P}% kein TextTpx --> P
210 }%
```

Um ein Textobjekt abzuschließen wird zunächst das Elternelement ermittelt. Dies liegt normalerweise oben auf dem Stack. Ist dieser leer wird das Element direkt unter der Wurzel eingefügt.

```
211 {%
212 \ifTextActive%
213 \ifnum \@stackdepth <1\relax%
214 \xdef\ParentElem{\theStructTree}%
215 \xdef\ParentArray{Karray}%
216 %\PackageWarning{accessibility}{stackdepth<1}%
217 \else%
218 \accessReadTopStack\ParentStructElem\ParentElem\ParentArray%
219 \fi%
220 %\convertLanguageInCode{\languagename}%
221 \immediate \pdfobj useobjnum\theTextObjNum{<</Type /StructElem %
222 /P \ParentElem \space 0 R %
223 /C /Normal %
224 /K [\TextArray] %
225 /S /\TextType %
226 \space\LanguageCode>>}%
227 \pdfrefobj \theTextObjNum%
228 \expandafter\xdef\csname \ParentArray\endcsname{%
229 \csname \ParentArray\endcsname \space \theTextObjNum\space 0 R}%
230 \global\TextActivefalse%
231 \fi%
232 }
```

PDFSpezialTextObj Im Unterschied zu normales Textobjekten sind besondere Textelemente im \LaTeX -Code speziell ausgezeichnet. Eine Erkennung ist also gewissermaßen zuverlässiger möglich. Da es sich bei speziellen Textobjekten auch um z. B. Formeln handeln kann, werden noch Variablen zur Attributverwaltung eingeführt werden.

```
233 \xdef\altAttr{}%
234 \xdef\titleAttr{}%
```

Der Beginn eines speziellen Textobjektes ist analog dem eines Normalen.

Parameter #1 StructType

```
235 \newenvironment*{PDFSpezialTextObj}[1]{%
236   \ifTextActive \endPDFMarkContent\endPDFTextObj\fi%
237   %neues anfangen
238   \global\SpezialTextActivetrue%
239   \pdfobj reserveobjnum% Objektnummer reservieren
240   \setcounter{TextObjNum}{\pdflastobj}%
241   \xdef\TextArray{}%
242   \xdef\TextType{#1}%
243 }%
```

Auch das Ende ist bis auf die Verwaltung der Attribute ähnlich. Eine Unterscheidung ist jedoch für spätere Zwecke nötig.

```
244 {%
245   %\gdef\LanguageHelp{}%
246   %\ifLanguageDiff%
247   %   \gdef\LanguageHelp{\LanguageCode}%
248   %\fi%
249   % \convertLanguageInCode{\languagesname}%
250   \gdef\AltHelp{}%
251   \ifthenelse{equal{\altAttr}{}}{}{}%
252   \gdef\AltHelp{/Alt(\altAttr)}%
253 }%
254 \gdef\TitleHelp{}%
255 \ifthenelse{equal{\titleAttr}{}}{}{}%
256 \gdef\TitleHelp{/T(\titleAttr)}%
257 }%
258 \ifnum \@stackdepth <1\relax%
259   \xdef\ParentElem{\theStructTree}%
260   \xdef\ParentArray{Karray}%
261 \else%
262   \accessReadTopStack\ParentStructElem\ParentElem\ParentArray%
263 \fi%
264 \immediate \pdfobj useobjnum\theTextObjNum{<</Type /StructElem %
265   /P \ParentElem \space 0 R %
266   /C /Normal %
267   /K [\TextArray] %
268   /S /\TextType %
269   \space\LanguageCode %
270   \space\TitleHelp %
271   \space\AltHelp>>}%
272 \pdfrefobj \theTextObjNum%
273 \expandafter\xdef\csname \ParentArray\endcsname{%
274   \csname \ParentArray\endcsname \space \theTextObjNum\space 0 R}%
275 \global\SpezialTextActivefalse%
276 \xdef\TextType{}%
277 \EveryparReset%
278 \xdef\altAttr{}% wieder leeren
279 \xdef\titleAttr{}% wieder leeren
280 }
```

3.2.4 Elemente auf Zeilenebene

`PDFInlineObjInText` Treten Objekte auf Zeilenebene in Textfluss auf, so muss dieser unterbrochen werden, das Objekt geschrieben werden und anschließend ist der Textfluss fortzusetzen. Zu diesem Zweck müssen einige Zustandvariablen gespeichert werden.

```
281 \xdef\lastEveryparType{}%
282 \xdef\HelpBool{}%
283 \xdef\Type{}%
284 \newcounter{PDFReferenceObjNum}%
285 \xdef\ReferenceArray{}%
```

Zuerst werden die alten Variablen gesichert. Anschließend ist für den Fall, dass es sich um Referenzen handelt, eine Sonderbehandlung nötig. Ihnen ist zusätzlich das von `hyperref` erzeugte Linkobjekt zu zuordnen. Da im Textfluss `\everypar` nicht greift, muss die Markierung des ContentStreams manuell vorgenommen werden.

```
286 \newenvironment*{PDFInlineObjInText}[1]{%
287   %alte einstellung merken
288   \xdef\lastEveryparType{\everyparStructElem}%
289   \xdef\HelpBool{\InlineObj}%
290   \xdef\Type{#1}%
291   \ifthenelse{equal{Type}{Reference} \or equal{Type}{Link}}{%
292     \pdfobj reserveobjnum%
293     \setcounter{PDFReferenceObjNum}{\pdflastobj}%
294     \setcounter{ObjNum}{\theTaggedObj}%
295     \EveryparConfig{#1}{obj}%
296   }{%
297     \EveryparConfig{#1}{true}%
298   }%
299   \PDFMarkContent% kein everypar
300 }
```

Anschließend ist die Markierung wieder zu beenden. Für Referenzen und Links muss nun die OBJR mit in die Struktur eingebunden werden. Das funktioniert über `pdflastlink`, aber erst ab PDFTEX Version 1.4.. Zum Schluss wird der Ausgangszustand wiederhergestellt und der nachfolgende Textfluss markiert.

```
301 {%
302   \endPDFMarkContent%
303   \ifthenelse{equal{Type}{Reference} \or equal{Type}{Link}}{%
304     \xdef\ReferenceArray{<</Type /MCR /Pg \pdfpageref\thepage %
305       \space \space 0 R /MCID \theObjNum>>}%
306     \ifthenelse{\pdftexversion>139}{%
307       \setcounter{ObjHelp}{\pdflastlink}%%<</Type /OBJR /Obj 600 0 R>>
308       %\PackageWarning{accessibility}%
309       %{Objektnummer vom letzten Link: \theObjHelp}%
310     }{\ifthenelse{\theObjHelp>0}{%
311       \xdef\ReferenceArray{\ReferenceArray\space %
312         <</Type /OBJR /Obj \theObjHelp\space 0 R>>}%
313     }}%
```

```

314 }{}%
315 \writeComplexTextObj{\thePDFReferenceObjNum}%
316     {\ReferenceArray}%
317     {/Reference}{\theTextObjNum}{NoPage}%
318 \xdef\TextArray{\TextArray \theObjHelp\space 0 R \space}%
319 }{}%
320 % alte einstellung wiederherstellen
321 \EveryparConfig{\lastEveryparType}{\HelpBool}%
322 \PDFMarkContent%
323 }

```

3.2.5 Marked Content

Zusätzlich zum Schreiben der Objekte müssen die zugehörigen Textpassagen im ContentStream markiert werden. Diese Funktionalität deckt das Makro PDFMarkContent ab.

Die Hilfsmakros `\EveryparConfig` und `\EveryparReset` dienen der Flexibilisierung von Mark-Content, indem sie dort verwendete Variablen setzen. `MarkContent` ist die Funktion die durch `\everypar` aufgerufen wird.

`\EveryparConfig` Setzt die Variablen auf die übergebenen Werte

Parameter #1 EveryparStructElem #2 true/false/obj

```

324 \newcommand{\EveryparConfig}[2]{%
325     \xdef\everyparStructElem{#1}%
326     \xdef\InlineObj{#2}%
327 }

```

`\EveryparReset` Setzt die Variablen auf die Standardwerte für eine normale Texterkennung.

```

328 \newcommand{\EveryparReset}{%
329     \xdef\everyparStructElem{P}%
330     \xdef\InlineObj{false}%
331 }

```

`PDFMarkContent` Zunächst werden wieder einige Variablen benötigt. Zum einen muss eine Markierung immer erst beendet werden, bevor eine neue angefangen wird. Der Sicherung dient die Variable `MarkContentOpen`. Auch die Markierung enthält einen Typ, der im allgemeinen denen des Textobjektes entspricht.

Die erzeugten Markierungen werden anschließend dem Strukturbaum zugeordnet. Dabei ist eine direkt Einbindung als `MarkContentReferenz` möglich, oder es kann ein Blattknoten erzeugt werden, der die Verbindung herstellt. Die Unterscheidung wird mittels `InlineObj` vorgenommen. Sie kann bisher die Belegungen `true`, `false` und `obj` annehmen.

```

332 \newif\ifMarkContentOpen \MarkContentOpenfalse%
333 \xdef\everyparStructElem{P}%
334 \xdef\InlineObj{false}%

```

Zunächst wird eine eventuell geöffnete Markierung abgeschlossen. Anschließend wird eine neue Markierung unter Verwendung der Funktion `\pdfliteral` in den ContentStream eingefügt. Da der Anfang einer nötigen Markierung zuverlässiger zu Erkennen ist, als das Ende wird die Verbindung zum Strukturbaum gleich anschließend durchgeführt. Des Weiteren werden Schachtelungsprobleme vermieden, bei denen die Zuordnung der MCID durch Überschreibung des `TaggedObj` ungültig wird.

```

335 \newenvironment{PDFMarkContent}{%
336   \ifMarkContentOpen \endPDFMarkContent\fi%
337   \global\MarkContentOpentrue%
338   \ifthenelse{equal{everyparStructElem}{}}{\EveryparReset}{}%
339   \pdfliteral{/everyparStructElem\space <</MCID \theTaggedObj>> BDC}%
340   \ifthenelse{equal{InlineObj}{false}}{% Text
341     %\ifthenelse{equal{themypage}{\thepage}}{}{}%
342     %\PackageWarning{accessibility}{unterschiedlich %
343     %\thepage<>\themypage; MCID \theTaggedObj}
344     }%
345     \xdef\TextArray{\TextArray \space <</Type /MCR %
346     %/Pg \pdfpageref\thepage \space \space 0 R %
347     /Pg \pdfpageref\themypage \space \space 0 R %
348     /MCID \theTaggedObj>>}%
349   }{}%
350   \ifthenelse{equal{InlineObj}{true}}{% inline-Objekt
351     \writePDFLeafObj{\theTaggedObj}{everyparStructElem}%
352   }{}%
353   \stepcounter{TaggedObj}%
354 }%
```

Wie bereits gesagt, sind die Enden von Absätzen nicht wirklich zuverlässig erkennbar, so dass häufig erst bei dem Beginn einer neuen Struktur auf ein Ende geschlossen werden kann. Eine frühest mögliche Erkennung ist insbesondere an Stellen des Seitenumbruchs relevant. Das Ende soll nach Möglichkeit noch auf der alten Seite auftauchen und nicht erst mit Beginn der neuen Struktur am Beginn der neuen Seite. Eine Verwendung von `\endMarkContent` ist relativ bedenkenlos mehrfach möglich, dafür dass nicht zu oft beendet wird, sorgt die Abfrage.

```

355 {%
356   \ifMarkContentOpen%
357     \pdfliteral{EMC}%
358     \global\MarkContentOpenfalse%
359   \fi%
360 }
```

Zur besseren Übersichtlichkeit und wurde das Makro `\writePDFLeafObj` ausgelagert. Es kann so des Weiteren auch an anderer Stelle wiederverwendet werden.

`\writePDFLeafObj` Diese Makro ermöglicht die Erzeugung einfacher Blattknoten, also derjenigen, die selbst keine Elemente mehr enthalten. Sie referenzieren die `/MCID` des zugehörigen Textabschnittes, diese wird normalerweise im Zähler `TaggedObj` gehalten, für einige Elemente

muss jedoch von dieser Nummerierung abgewichen werden, die Variable in im ersten Argument und der Strukturtyp im zweiten zu übergeben.

Parameter #1 MCID #2 Structure

```

361 \newcommand*{\writePDFLeafObj}[2]{%
362   \%gdef\LanguageHelp{}%
363   \%ifLanguageDiff%
364   \%   \gdef\LanguageHelp{\LanguageCode}%
365   \%fi%
366   \% \convertLanguageInCode{\languageName}%
367   \gdef\AltHelp{}%
368   \ifthenelse{equal{\altAttr}{}}{ }{%
369     \gdef\AltHelp{/Alt(\altAttr)}%
370   }%
371   \gdef\TitleHelp{}%
372   \ifthenelse{equal{\titleAttr}{}}{ }{%
373     \gdef\TitleHelp{/T(\titleAttr)}%
374   }%
375   \immediate \pdfobj {<</Type /StructElem %
376     /P \theTextObjNum \space 0 R %
377     /C /Normal %
378     /K <</Type /MCR %
379       /Pg \pdfpageref\thepage \space \space 0 R %
380       /MCID #1>> %
381     /S /#2 %
382     \space\LanguageCode %
383     \space\TitleHelp %
384     \space\AltHelp>>}%
385   \setcounter{ObjHelp}{\pdfLastObj}%
386   \pdfrefobj \pdfLastObj%
387   \xdef\TextArray{\TextArray \space \theObjHelp\space 0 R}%
388   \xdef\altAttr{}%
389   \xdef\titleAttr{}%
390 }

```

`\writeComplexTextObj` Komplexere Objekte wie die von Fußnoten, in denen zusätzlich die Strukturen am Seitenende integriert werden, sind mit `\writePDFLeafObj` nicht zu realisieren. Diese Funktionalität bietet dieses Makro. Es sind ermöglicht die Erzeugung von mittleren Knoten, also denjenigen, die selbst weitere Objekte enthalten. Damit die doppelte Verlinkung (Eltern mit Kindern, wie umgekehrt) möglich ist, muss bei der Erzeugung der Kinderelemente die Objektzahl des Elternobjektes bekannt sein. Des Weiteren wird jeweils die Referenz des Kindes in die Kinderliste der Eltern eingefügt. So dass dieses erst nach der Generierung aller Kinder erzeugt werden kann. Zur Realisierung wird am Anfang einer mittleren Struktur eine Objektzahl reserviert, die im ersten Parameter übergeben.

Die Erzeugung von Objekten kann sowohl mit, als auch ohne Seitenreferenz erfolgen. Zudem muss die Liste der Kinder, das Elternelement und der Strukturtyp an das Makro übergeben werden.

Parameter #1 Objektnummer #2 KidsArray #3 StrukturType #4 ParentObj #5 (no)page

```
391 \newcommand*{\writeComplexTextObj}[5]{%
392   \ifthenelse{\equal{#5}{Page}}{% Seitenreferenz angeben
393     \immediate \pdfobj useobjnum#1{<</Type /StructElem %
394       /P #4\space 0 R %
395       /Pg \pdfpageref\thepage \space \space 0 R %
396       /C /Normal %
397       /K [#2] %
398       /S #3 \space\LanguageCode>>}}%
399   }{%keine Seitenreferenz angeben
400     \immediate \pdfobj useobjnum#1{<</Type /StructElem %
401       /P #4\space 0 R %
402       /C /Normal %
403       /K [#2] %
404       /S #3>>}}%
405   }%
406   \setcounter{ObjHelp}{\pdflastobj}%
407   \pdfrefobj \pdflastobj%
408 }
```

3.3 Erkennen von Absätzen

Da Textabsätze in L^AT_EX nicht explizit ausgezeichnet sind, wird `\everypar` verwendet um den Anfang eines Textabschnittes zu erkennen. Sobald eine andere Struktur erkannt wird, wird der Textabschnitt beendet. In längeren Strukturen wird die Absatzerkennung mit `\everypar` explizit ausgesetzt und nach Abschluss wieder aktiviert.

`numberingpars` dient der Unterscheidung, ob Absatznummern generiert werden sollen oder eben nicht. Während `npar` von jedem `\everypar` zurückgesetzt wird und bestimmt, ob der folgende Absatz nummeriert sein soll. In Kombination ergeben beide ein Werkzeug zur Entscheidung, ob `\everypar` nun einen wirklichen Textabsatz einleitet.

Der Zähler `ParCounter` dient nur zu Testzwecken, er zählt die wirklichen Textabsätze je Seite mit. Es ist z. B. möglich auskommentierten Zeilen zu aktivieren, um z sehen an welchen Stellen `\everypar` greift.

```
409 %\newcounter{ParCounter}[page]%
410 \newcounter{ParCounter}%
411 \newif\ifnumberingpars \numberingparstrue%
412 \newif\ifnpar \npartrue%
```

Nun wird die Funktion `\everypar` tatsächlich undefiniert. Dazu werden zunächst die ursprüngliche Definition in `\originaleverypar` sowie die Token der Funktion gesichert. Anschließend wird `\everypar` unter Verwendung der Originaldefinition erweitert. Ist nun die Textabsatzerkennung bestehend aus `numberingpars` und `npar` aktiviert. Löst `\everypar` den Start der Umgebung `\PDFText` aus. Sofern diese noch aktiv ist, wird sie zuvor beendet.

```
413 \ifthenelse{\boolean{@Access@pdf}}{%
414   \let\originaleverypar\everypar%
```

```

415 \newtoks\npeverypar%
416 \npeverypar{}%
417 % Call everypar with the argument extended with the toks
418 \def\everypar#1{%
419   \originalevypar{#1\ifnumberingpars\the\npeverypar\fi}}%
420 \npeverypar{%
421   \ifnpar{%
422     \stepcounter{ParCounter}%
423     %\pdfliteral{/Artifact BMC}%
424     % \llap{\small\arabic{ParCounter}\quad}%
425     %\pdfliteral{EMC}%
426     \ifSpezialTextActive\else \PDFTextObj \fi%
427     \PDFMarkContent%
428   }\fi%
429 }%
430 }{}

```

3.4 Dokumentbeginn

Strukturbaum eröffnen.

```

431 \AtBeginDocument{%
432 \PDFStructObj{Document}{\empty}%
433 \everypar{}%
434 }

```

3.5 Dokumentende

Strukturbaum abschließen

```

435 \ifthenelse{\boolean{@Access@pdf}}{%
436   \AtEndDocument{%

```

Am Ende des Dokumentes müssen alle noch offenen Strukturen abgeschlossen werden. Diesen Zweck erfüllen die folgenden Zeilen.

```

437   \closeUntilPDFStruct{Document}%

```

Um in späteren Implementierungen das „Umfließen“ zu unterstützen muss jedem Strukturelement eine Layout-Klasse zugeordnet werden. Das Klassenzuordnungsobjekt verwaltet diese, in dem es im einfachsten Fall einem Klassennamen (/Normal) verschiedene Layoutattribute zuordnet(/TextAlign /Center). Bisher verhindert die klare Wortbegrenzung (fehlende Leerzeichen) im PDF-Quellcode des ContentStreams ein automatisches Reflow.

```

438   \newcounter{ClassMap}%
439   \pdfobj reserveobjnum% neues Objekt Reservieren
440   \setcounter{ClassMap}{\pdfastobj}%
441   \immediate \pdfobj useobjnum \theClassMap{<<%
442     /Normal <</O /Layout /EndIndent 0.0 %
443     /SpaceAfter 0.0 /SpaceBefore 0.0 %
444     /StartIndent 0.0 /WritingMode %

```

```

445         /LrTb /TextAlign /Start>> %
446     /CM1 <</O /Layout /TextAlign /Justify>> %
447     /CM2 <</O /Layout /TextAlign /Center>> %
448     /CM3 <</O /Layout /TextAlign /Start>> %
449     /CM4 <</O /Layout /InlineAlign /Center %
450         /Placement /Block /SpaceAfter 12.125 %
451         /BBox [266 314 329 336]>> %
452     >>} \pdfrefobj \pdflastobj%
453     %\global\setcounter{ClassMap}{\pdflastobj}%

```

Des Weiteren wurden im Verlauf der Abarbeitung eigene PDF-Strukturen abgeleitet. Sie müssen nun einem Standard-Element zugeordnet werden. Dazu wird das so genannte Rollenzuordnungsobjekt geschrieben. Es ordnet jeweils die selbst definierte (`/IndexItem`) einer Standardrolle (`/Span`) zu.

```

454 \pdfobj {<<%
455     /IndexItem /Span % Indexeinträge
456     /TOF /TOC % Table of Figures
457     /TOFI /TOCI % Table of Figures Eintrag
458     /TOT /TOC % Table of Tables
459     /TOTI /TOCI % Table of Tables Eintrag++
460     /Titlepage /Sect % Titlepage
461     /Bibliography /L % Bibliography
462     /BibItem /LI % BibliographyItem
463     /ParagraphSpan /Span % geteilte Paragraphen am Seitenumbruch
464     /Footnote /Note % Fußnotentext
465     /Chapter /Sect%
466     /Section /Sect%
467     /Subsection /Sect%
468     /Subsubsection /Sect%
469     /Float /Div%
470     /L1 /L%
471     /L2 /L%
472     /L3 /L%
473     /L4 /L%
474     /L5 /L%
475     >>} \pdfrefobj \pdflastobj%
476 \setcounter{ObjHelp}{\pdflastobj}%

```

Erst jetzt, wenn alle Objekte ins PDF-Dokument geschrieben wurden kann, dass Strukturwurzelobjekt erzeugt werden, da es Referenzen auf die anderen Objekte enthält.

```

477 \immediate \pdfobj useobjnum \theStructTree%
478     <</Type /StructTreeRoot %
479     /RoleMap \theObjHelp \space 0 R %
480     /ClassMap \theClassMap \space 0 R %
481     /ParentTree <</Nums [0 [\Karray]]>> % TODO Viel komplizierter
482     /ParentTreeNextKey 1 % berechnen
483     /K [\Karray] %
484     >>}\pdfrefobj\pdflastobj%

```

Nun wird der gesamte erzeugte Strukturbaum in den Katalog der PDF-Datei eingefügt und das PDF als „Tagged PDF“ ausgewiesen.

```
485 \pdfcatalog{%
486   /StructTreeRoot \theStructTree\space 0 R%
487   /MarkInfo <</Marked true /LetterspaceFlags 0>>%
488   %/PieceInfo<</MarkedPDF>>%
489   %/MarkInfo <</Marked true>>%
490   %/Metadata \theStructTree\space 0 R%
491 }%
```

Abschließend wird dem Autor, mitgeteilt, ob und wo noch Probleme bzgl. der Barrierefreiheit bestehen.

```
492 \ifthenelse{\boolean{ACCESSProblems}}{%
493   \PackageWarningNoLine{accessibility}{%
494     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
495     @@ There were non-defined Figure Alt-Tags! @@\MessageBreak
496     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@}%
497 }{}%
498 }%
499 }{}
```

3.6 Seitenumbruch

Bisher ist eine korrekte Auszeichnung der Seitenumbrüche nicht möglich, da die Autorin bisher keine Variante zur zuverlässigen Erkennung gefunden hat.

Prinzipiell müsste an jedem Seitenende (vor dem Schreiben der Fußnoten) geschaut werden, welche Struktur im ContentStream noch aktiv ist, dass betrifft prinzipiell Elemente wie /P, /Lbody, etc., also Objekte die als Blattobjekte mit einer Seitenreferenz geschrieben werden müssen. Diese müssen durch ein `\pdfliteral{EMC}` unterbrochen werden. Damit wäre immerhin die Integrität des ContentStreams gewährleistet.

3.6.1 Automatischer Seitenumbruch

An die richtige Stelle der letzten Seite kann das Literal z.B. mittels `\@textbottom` gesetzt werden. Es wird während der Ausgabe (`\shipout`) aufgerufen. Allerdings sieht dies nicht genau zum Seitenumbruch, sondern erst nach ein bis zwei Absatzboxen. Damit sind die Variablen wie `ifPDFTextActiv` nicht mehr aktuell und eine Erkennung ist nicht wirklich zuverlässig möglich.

Auf der neuen Seite muss die Struktur dann natürlich wieder geöffnet werden. Dann müssen beide Teile mit der richtigen Seitenreferenz als Element in den Baum einsortiert werden. Beide Funktionen können Inhalte auf der gerade fertiggestellten Seite hinzufügen.

Fußnoten werden im Moment ihres Auftauchens in eine temporäre Box geschrieben und später in den Output eingefügt. `\@texttop` wird immer vor `\@textbottom` durch `\shipout` aufgerufen.

Die folgende Implementierung funktioniert, aber nur in einem von 3 Spezialfällen. Dies ist eindeutig noch eine Baustelle.

```

500 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
501 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
502 \newcount\linenopenalty\linenopenalty=-100000%
503 \mathchardef\linenopenaltypar=32000%
504 %
505 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Verändern der Outputroutine%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
506 \ifthenelse{boolean{Access@pdf}}{%
507   \let\@tempa\output%
508   \newtoks\output%
509   \let\@AC@output\output%
510   \output=\expandafter{\the\@tempa}%
511   %
512   \@tempa={%
513     % LineNoTest
514     \let\@@par\@@par%
515     \ifnum\interlinepenalty<-\linenopenaltypar%
516       \advance\interlinepenalty-\linenopenalty%
517       \@AC@nobreaktrue%
518     \fi%
519     \@tempswatrue%
520     \ifnum\outputpenalty>-\linenopenaltypar\else%
521       \ifnum\outputpenalty>-188000\relax%
522         \@tempswafalse%
523       \fi%
524     \fi%
525     \if@tempswa%
526       % LineNoLaTeXOutput
527       \ifnum \holdinginserts=\thr@@ %
528         \global\holdinginserts-\thr@@ %
529         \unvbox\@cclv %
530         \ifnum \outputpenalty=\@M \else \penalty\outputpenalty \fi %
531       \else%
532         \if@twocolumn \let\@makecol\@AC@makecol \fi%
533         \the\@AC@output %
534         \ifnum \holdinginserts=-\thr@@ %
535           \global\holdinginserts\thr@@ \fi %
536         \fi%
537       \else %
538         %MakeLineNo
539         \boxmaxdepth\maxdimen\setbox\z@\vbox{\unvbox\@cclv}%
540         \@tempdima\dp\z@ \unvbox\z@%
541         \sbox\@tempboxa{\hb@xt@\z@{\makeLineNumber}}%
542         \stepcounter{linenumber}%
543         \stepcounter{abslinenumber}%
544         \ht\@tempboxa\z@ \@AC@depthbox %
545         \count@\lastpenalty %
546         \ifnum\outputpenalty=-\linenopenaltypar %
547         \ifnum\count@=\z@ \else %

```

```

548         \xdef\@AC@parpgbrk{%
549             \penalty\the\count@
550             \global\let\noexpand\@AC@parpgbrk%
551                 \noexpand\@AC@screenoff@pen}%
552         \fi%
553     \else%
554         \@tempcnta\outputpenalty%
555         \advance\@tempcnta -\linenopenalty%
556         \penalty \ifnum\count@<\@tempcnta \@tempcnta \else \count@ \fi %
557     \fi%
558 \fi%
559 }%
560 \def\@AC@nobreaktrue{\let\if@nobreak\iftrue} %
561 %
562 \def\@AC@depthbox{%
563     \dp\@tempboxa=\@tempdima%
564     \nointerlineskip \kern-\@tempdima \box\@tempboxa} %
565 %
566 \def\@AC@screenoff@open{%
567     \ifdim\lastskip=\z@ %
568         \@tempdima\prevdepth \setbox\@tempboxa\null %
569         \@AC@depthbox \fi}%
570 %
571 \global\let\@AC@parpgbrk\@AC@screenoff@open %
572 }-}%
573 %
574 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Initializieren der Variablen%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
575 %
576 %Zeilennummer
577 \newcounter{linenumber}%
578 \newcounter{abslinenumber}%
579 %Seitennummer
580 \newcount\c@AC@truepage %
581 \global\advance\c@AC@truepage\@ne %mit eins beginnen
582 %\g@addto@macro\cl@page{\global\c@AC@truepage\c@page}%
583 \g@addto@macro\cl@page{\global\advance\c@AC@truepage\@ne}%
584 \@addtoreset{AC@truepage}{\ckpt}%
585 %
586 \newcounter{mypage}%
587 \setcounter{mypage}{\@ne}%
588 \g@addto@macro\cl@page{\global\c@mypage\c@page}%
589 \@addtoreset{mypage}{\ckpt}%
590 %
591 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Schreiben der Zeilennummern%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
592 \ifthenelse{\boolean{Access@pdf}}{%
593     \def\makeLineNumber{%
594         \protected@write\auxout}{\string\@AC{\the\c@linenumber}%
595             {\noexpand\the\c@AC@truepage}}%
596         \testNumberedPage%
597         %Schreibt die Zeilennummern
598         %\hss{\normalfont\tiny\sffamily\thelinenumber\quad}%
599     }%

```

```

600 }{}%
601 %
602 %%%%%%%%%%%%%Absatz
603 \newif\ifLastLineStop \LastLineStopfalse%
604 \newcounter{LastPar}%
605
606 \newcommand{\EndPage}[1]{%
607   \ifMarkContentOpen%
608     \endPDFMarkContent%
609     \global\LastLineStoptrue%
610     \setcounter{LastPar}{\noexpand\theParCounter}%
611   \else%
612     \global\LastLineStopfalse%
613   \fi%
614   \if@twocolumn \else%
615     \stepcounter{mypage}%
616   \fi%
617 }%
618
619 \newcommand{\BeginPage}{%
620   \ifLastLineStop%
621     \ifnum \theParCounter=\theLastPar%
622       \pdfliteral{/P <</MCID \theTaggedObj>> BDC}%
623       \global\MarkContentOpentrue%
624       \xdef\TextArray{\TextArray \space <</Type /MCR %
625         /Pg \pdfpageref\the\c@AC@truepage \space \space 0 R %
626         /MCID \theTaggedObj>>}%
627       \stepcounter{TaggedObj}%
628     \fi%
629   \fi%
630 }%
631
632 \ifthenelse{\boolean{@Access@pdf}}{%
633   \let\original@startcolumn\startcolumn%
634   \renewcommand{\@startcolumn}{%
635     \BeginPage%
636     \original@startcolumn%
637   }%
638 }{}%
639
640 %%%%%%%%%%%%%Berichtigung der Seitennummer%%%%%%%%%%%%
641 \ifthenelse{\boolean{@Access@pdf}}{%
642   \def\LastNumberedPage{first}%
643   \def\AC@Pfirst{\nextAC\relax}%
644   %
645   \let\lastAC\relax % compare to last line on this page
646   \let\firstAC\relax % compare to first line on this page
647   \let\pageAC\relax % get the page number, compute the linenumber
648   \let\nextAC\relax % move to the next page
649   %
650   \AtEndDocument{\let\@AC\@gobbletwo} %
651   %

```

```

652 \def\AC#1#2{\expandafter\@AC%
653         \csname AC@P#2C\@AC@column\expandafter\endcsname%
654         \csname AC@P0#2\endcsname%
655         {#1}{#2}}}%
656 %
657 \def\@AC#1#2#3#4{\ifx#1\relax%
658     \ifx#2\relax\gdef#2{#3}\fi%
659     \expandafter\@@AC\csname AC@P\LastNumberedPage\endcsname#1%
660     \xdef#1{\lastAC{#3}\firstAC{#3}}%
661     \pageAC{#4}{\@AC@column}{#2}\nextAC\relax}%
662 \else%
663     \def\lastAC##1{\noexpand\lastAC{#3}}%
664     \xdef#1{#1}%
665     \fi%
666     \xdef\LastNumberedPage{#4C\@AC@column}%
667 }%
668 %
669 \def\@@AC#1#2{\def\nextAC##1{\noexpand\nextAC\noexpand#2}%
670     \xdef#1{#1}}}%
671 %
672 \def\NumberedPageCache{\AC@Pfirst}%
673 %
674 \def\testLastNumberedPage#1{%
675     \ifnum#1<\c@linenumber%
676         \let\firstAC@gobble%
677         \fi%
678     \ifnum#1=\c@linenumber%
679         \EndPage{#1}%
680         \fi%
681 }%
682 %
683 \def\testFirstNumberedPage#1{%
684     \ifnum#1>\c@linenumber%
685         \def\nextAC##1{\testNextNumberedPage\AC@Pfirst}%
686         \else%
687             \let\nextAC@gobble%
688             \def\pageAC{\gotNumberedPage{#1}}%
689             \fi%
690 }%
691 %
692 \long\def \@gobblethree #1#2#3{}%
693 %
694 \def\testNumberedPage{%
695     \let\lastAC\testLastNumberedPage%
696     \let\firstAC\testFirstNumberedPage%
697     \let\pageAC@gobblethree%
698     \let\nextAC\testNextNumberedPage%
699     \NumberedPageCache%
700 }%
701 %
702 \def\testNextNumberedPage#1{%
703     \ifx#1\relax%

```

```

704     \global\def\NumberedPageCache{\gotNumberedPage0000}%
705     \PackageWarning{accessibility}{Changed paragraphs, rerun to get it right}%
706     \else%
707         \global\let\NumberedPageCache#1%
708         \fi%
709     \testNumberedPage%
710 }%
711 %
712 \def\gotNumberedPage#1#2#3#4{%
713     \ifodd \if@twocolumn #3\else #2\fi\relax\fi%
714     \advance\c@linenumber\@ne % Nummerierung ab 1 sonst ab 0
715     \advance\c@linenumber-#4\relax%
716 }%
717 }-%
718 %
719 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Spaltenerkennung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
720 \ifthenelse{\boolean{Access@pdf}}{-%
721     \def\@AC@col{\def\@AC@column}%
722     \@AC@col{1}%
723     %
724     \AtBeginDocument{\let\@AC@orig@makecol\@makecol}%
725     %
726     \def\@AC@makecol{%
727         \@AC@orig@makecol
728         \setbox\@outputbox \vbox{%
729             \boxmaxdepth \@maxdepth%
730             \protected@write\@auxout{}-%
731                 \string\@AC@col{\if@firstcolumn1\else2\fi}%
732             }%
733         \box\@outputbox %
734     }-%
735 }-%
736 %
737 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Absatzerkennung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
738 \ifthenelse{\boolean{Access@pdf}}{-%
739     \let\@@@par\@@par%
740     \newcount\linenoprevgraf%
741     %
742     \def\linenumberpar{%
743         \ifvmode \@@@par \else %
744             \ifinner \@@@par \else%
745                 \xdef\@AC@outer@holdins{\the\holdinginserts}%
746                 \advance \interlinepenalty \linenopenalty%
747                 \linenoprevgraf \prevgraf%
748                 \global \holdinginserts \thr@@ %
749                 \@@@par%
750                 \ifnum\prevgraf>\linenoprevgraf%
751                     \penalty-\linenopenaltypar%
752                 \fi%
753                 \@AC@parpgbrk %
754                 \global\holdinginserts\@AC@outer@holdins%
755                 \advance\interlinepenalty -\linenopenalty%

```

```

756     \fi%
757     \fi}%
758 %
759 \AtEndOfPackage{%
760     \xdef\AC@outer@holdins{\the\holdinginserts}%
761     \let\@@par\linenumberpar%
762     \ifx\@par\@@par\let\@par\linenumberpar\fi%
763     \ifx\par\@@par\let\par\linenumberpar\fi%
764 }%
765 }{}%
766 %
767 %%%%%%%%%%%%%Formelbehandlung%%%%%%%%%%%%
768 \def\linenomath{%
769     \ifnum\interlinepenalty>-\linenopenaltypar
770         \global\holdinginserts\thr@@
771         \advance\interlinepenalty \linenopenalty
772         \ifhmode %
773             \advance\predisplaypenalty \linenopenalty
774             \fi
775             \advance\postdisplaypenalty \linenopenalty
776             \advance\interdisplaylinepenalty \linenopenalty
777             \fi
778             \ignorespaces
779 }%
780
781 \def\endlinenomath{%
782     \global\holdinginserts\AC@outer@holdins %
783     \global\@ignoretrue
784 }
785
786 \ifthenelse{\boolean{Access@pdf}}{%
787     \@ifundefined{mathindent}{%
788         \let\AC@displaymath\[%
789         \let\AC@enddisplaymath\]%
790         \renewcommand\{\begin{linenomath}\AC@displaymath}%
791         \renewcommand\{\AC@enddisplaymath\end{linenomath}}%
792         %
793         \let\AC@equation\equation%
794         \let\AC@endequation\endequation%
795         \renewenvironment{equation}%
796             {\linenomath\AC@equation}%
797             {\AC@endequation\endlinenomath}%
798     }{}%
799 %
800     \let\AC@eqnarray\eqnarray%
801     \let\AC@endeqnarray\endeqnarray%
802     \renewenvironment{eqnarray}%
803         {\linenomath\AC@eqnarray}%
804         {\AC@endeqnarray\endlinenomath}%
805 %
806     \advance\maxdeadcycles 100%
807 }{}%

```

```

808 %
809 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
810 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

3.6.2 Manueller Seitenumbruch

Auch mit den Befehlen `\pagebreak`, `\nopagebreak` und `\newpage` sollte der Algorithmus funktionieren.

Eventuell kann hier eine Möglichkeit geschaffen werden am Ende des Dokumentenerstellungsprozesses, wenn also alles fertig ist, nicht automatisch erkennbare Absätze per Hand zu kennzeichnen. Die Befehle `\clearpage` und `\cleardoublepage` greifen auf die Definition von `\newpage` zurück.

```

811 \ifthenelse{\boolean{@Access@pdf}}{%
812   \let\originalnewpage\newpage%
813   \renewcommand{\newpage}{%
814     \endPDFMarkContent%
815     \originalnewpage%
816   }%
817 }{}%

```

3.7 Überschriften

Die Latex-Struktur

```

...
\section{Überschrift}
Absatz...
\subsection{Unterüberschrift}
Absatz...
\subsection{Unterüberschrift}
...

```

Die PDF-Struktur

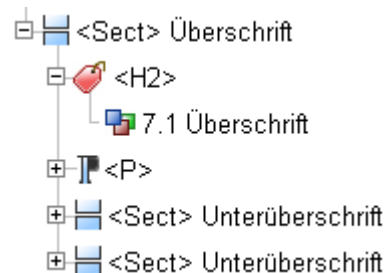


Abbildung 3.1: Struktur für Überschriften

3.7.1 Hilfsmakro

PDFSect Beginnt ein neues Strukturelement, aber nur in dem Fall, dass die Option `highstructure` gesetzt ist. Anschließend wird ein spezielles Textobjekt (H) begonnen, und die Absatzmarkierung konfiguriert.

```

818 \newenvironment{PDFSect}[2]{%
819   \ifPDFDetailedStructure%
820     \closeUntilPDFStruct{#1}%
821     \PDFStructObj{#1}{#2}%
822   \fi%

```



```

823 \PDFSpezialTextObj{H}%
824 \EveryparConfig{H}{false}%
825 }%

```

Am Ende der Überschrift wird nur die Markierung der Textpassage und das Textobjekt beendet. Die Struktur beginnt ja mit der erst. Sie wird bei Beginn einer höherliegenden Gliederungsebene geschlossen.

```

826 {%
827 \endPDFMarkContent%
828 \endPDFSpezialTextObj%
829 }

```

Nachdem nun die abstrakten Hilfsmakros angelegt sind, können die betroffenen Gliederungsbe-
fehle umdefiniert werden.

3.7.2 Kapitel

Da der Gliederungsbefehl für Kapitel nur in einigen Dokumentenklassen angeboten wird, ist hierzu eine Sonderbehandlung nötig. Für die verschiedenen Aufrufe ist zudem eine Fallunter-
scheidung nötig.

Umdefinieren des chapter-Befehls

`chapter` Das Umdefinieren des `\chapter`-Befehls.

```

830 \ifthenelse{\boolean{@Access@pdf}}{%
831 \ifundefined{chapter}{% es gibt keine Chapter z.B. in Article-Klassen
832 }{%
833 \let\originalchapter\chapter%
834 \renewcommand{\chapter}{%Sortiert die verschiedenen Aufrufe
835 \@ifstar{\originalchapterWithStar}%\chapter*{Beispielkapitel}
836 {\@ifnextchar[%]
837 {\originalchapterWithTwoOption}%\chapter[BspKap]{Beispielkapitel}
838 {\originalchapterWithOption}%\chapter{Beispielkapitel}
839 }%
840 }%
841 }%
842 }{}

```

Zuordnung der verschiedenen Aufrufvarianten.

```

843 \newcommand{\originalchapterWithStar}[1]{%
844 \PDFSect{Chapter}{#1}\originalchapter*{#1}\endPDFSect}%
845 \newcommand{\originalchapterWithTwoOption}[2]{%
846 \PDFSect{Chapter}{#1}\originalchapter[#1]{#2}\endPDFSect}%
847 \newcommand{\originalchapterWithOption}[1]{%
848 \PDFSect{Chapter}{#1}\originalchapter{#1}\endPDFSect}%

```

addchap Das Umdefinieren des `\addchap`-Befehls.

```
849 \ifthenelse{\boolean{@Access@pdf}}{%
850   \@ifundefined{addchap}{% es gibt keine Chapter z.B. in Article-Klassen
851   }{%
852     \let\originaladdchap\addchap%
853     \renewcommand{\addchap}{%
854       \@ifstar{\originaladdchapWithStar}%
855       {\@ifnextchar[%
856         {\originaladdchapWithTwoOption}%
857         {\originaladdchapWithOption}%
858       }%
859     }%
860   }%
861 }{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
862 \newcommand{\originaladdchapWithStar}[1]{%
863   \PDFSect{Chapter}{#1} \originaladdchap*{#1} \endPDFSect}%
864 \newcommand{\originaladdchapWithTwoOption}[2]{%
865   \PDFSect{Chapter}{#1} \originaladdchap[#1]{#2} \endPDFSect}%
866 \newcommand{\originaladdchapWithOption}[1]{%
867   \PDFSect{Chapter}{#1} \originaladdchap{#1} \endPDFSect}%

```

Im KOMA-Script gibt es die Möglichkeit eine Präamble für Kapitel und Parts zu setzen. Diese wird durch die nächsten Zeilen als `/P` ausgezeichnet.

```
868 \ifthenelse{\boolean{@Access@pdf}}{%
869   \@ifundefined{set@preamble}{% es gibt kein set@preamble%
870   }{% %außerhalb des KOMA-Scripts
871     \let\originaluse@preamble\use@preamble%
872     \renewcommand{\use@preamble}[1]{%
873       \EveryparConfig{P}{true}%
874       \originaluse@preamble{#1}%
875       \EveryparConfig{H}{false}%
876     }%
877   }%
878 }{}
```

3.7.3 Überschriften mit `Afterskip`

Diese Gliederungsebenen gibt es in allen Dokumentenklassen.

section Umdefinieren des `\section`-Befehls

```
879 \ifthenelse{\boolean{@Access@pdf}}{%
880   \let\originalsection\section%
881   \renewcommand{\section}{%
882     \@ifstar{\originalsectionWithStar}%

```

```

883   {\@ifnextchar[%]
884     {\originalsectionWithTwoOption}%
885     {\originalsectionWithOption}%
886   }%
887 }%
888 }{}

```

Zuordnung der verschiedenen Aufrufvarianten.

```

889 \newcommand{\originalsectionWithStar}[1]%
890   {\PDFSect{Section}{#1} \originalsection*{#1} \endPDFSect}%
891 \newcommand{\originalsectionWithTwoOption}[2]%
892   {\PDFSect{Section}{#1} \originalsection[#1]{#2} \endPDFSect}%
893 \newcommand{\originalsectionWithOption}[1]%
894   {\PDFSect{Section}{#1} \originalsection{#1} \endPDFSect}%

```

subsection Umdefinieren des `\subsection`-Befehls

```

895 \ifthenelse{\boolean{@Access@pdf}}{%
896   \let\originalsubsection\subsection%
897   \renewcommand{\subsection}{%
898     \@ifstar{\originalsubsectionWithStar}%
899     {\@ifnextchar[%]
900       {\originalsubsectionWithTwoOption}%
901       {\originalsubsectionWithOption}%
902     }%
903   }%
904 }{}

```

Zuordnung der verschiedenen Aufrufvarianten.

```

905 \newcommand{\originalsubsectionWithStar}[1]%
906   {\PDFSect{Subsection}{#1} \originalsubsection*{#1} \endPDFSect}%
907 \newcommand{\originalsubsectionWithTwoOption}[2]%
908   {\PDFSect{Subsection}{#1} \originalsubsection[#1]{#2} \endPDFSect}%
909 \newcommand{\originalsubsectionWithOption}[1]%
910   {\PDFSect{Subsection}{#1} \originalsubsection{#1} \endPDFSect}%

```

subsection Umdefinieren des `\subsubsection`-Befehls

```

911 \ifthenelse{\boolean{@Access@pdf}}{%
912   \let\originalsubsubsection\subsubsection%
913   \renewcommand{\subsubsection}{%
914     \@ifstar{\originalsubsubsectionWithStar}%
915     {\@ifnextchar[%]
916       {\originalsubsubsectionWithTwoOption}%
917       {\originalsubsubsectionWithOption}%
918     }%
919   }%
920 }{}

```

Zuordnung der verschiedenen Aufrufvarianten.

```
921 \newcommand{\originalsubsubsectionWithStar}[1]%
922   {\PDFSect{Subsubsection}{#1} \originalsubsubsection*{#1} \endPDFSect}%
923 \newcommand{\originalsubsubsectionWithTwoOption}[2]%
924   {\PDFSect{Subsubsection}{#1} \originalsubsubsection[#1]{#2} \endPDFSect}%
925 \newcommand{\originalsubsubsectionWithOption}[1]%
926   {\PDFSect{Subsubsection}{#1} \originalsubsubsection{#1} \endPDFSect}%
```

3.7.4 Überschriften ohne Afterskip

In der im `scrrept`-Definierten Überschriftsvariante werden `\paragraph` und `\subparagraph` ohne nachfolgenden Zeilenumbruch gesetzt. Solche Überschriften werden als Textabschnitt gekennzeichnet.

`PDFParagraphSect` Nachdem wieder ein Strukturobjekt erzeugt wurde. Beginnt `\PDFTextObj` ein normales TextObjekt. Die Markierung des ContentStreams muss in diesem Fall explizit geöffnet werden, da die Überschrift durch `\everypar` vor den Absatz gesetzt wird und somit nicht richtig erkannt wird.

```
927 \newenvironment{PDFParSect}[2]{%
928   %\ifPDFDetailedStructure%
929   %   \closeUntilPDFStruct{#1}%
930   %   \PDFStructObj{#1}{#2}%
931   %\fi%
932   \PDFTextObj%
933   \EveryparConfig{P}{false}%
934   \PDFMarkContent%
935 }%
```

Die Erkennung des Endes kann `\everypar` aber durchaus überlassen werden. An dieser Stelle wäre die Beendigung zu früh und würde zu einer leeren Markierung führen.

```
936 {%
937   %\endPDFMarkContent% erst durch everypar
938   %\endPDFTextObj%
939 }
```

`paragraph` Umdefinieren des `\paragraph`-Befehls

```
940 \ifthenelse{\boolean{@Access@pdf}}{%
941   \let\originalparagraph\paragraph%
942   \renewcommand{\paragraph}{%
943     \ifstar{\originalparagraphWithStar}%
944     {\@ifnextchar[%
945       {\originalparagraphWithTwoOption}%
946       {\originalparagraphWithOption}%
947     }%
948   }
```

```
948 }%
949 }{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
950 \newcommand{\originalparagraphWithStar}[1]%
951   {\PDFParSect{Paragraph}{#1} \originalparagraph*{#1} \endPDFParSect}%
952 \newcommand{\originalparagraphWithTwoOption}[2]%
953   {\PDFParSect{Paragraph}{#1} \originalparagraph[#1]{#2} \endPDFParSect}%
954 \newcommand{\originalparagraphWithOption}[1]%
955   {\PDFParSect{Paragraph}{#1} \originalparagraph{#1} \endPDFParSect}%
```

subparagraph Umdefinieren des `\subparagraph`-Befehls

```
956 \ifthenelse{\boolean{@Access@pdf}}{%
957   \let\originalsubparagraph\subparagraph%
958   \renewcommand{\subparagraph}{%
959     \@ifstar{\originalsubparagraphWithStar}%
960     {\@ifnextchar[%]
961       {\originalsubparagraphWithTwoOption}%
962       {\originalsubparagraphWithOption}%
963     }%
964   }%
965 }{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
966 \newcommand{\originalsubparagraphWithStar}[1]%
967   {\PDFParSect{Subparagraph}{#1} \originalsubparagraph*{#1} \endPDFParSect}%
968 \newcommand{\originalsubparagraphWithTwoOption}[2]%
969   {\PDFParSect{Subparagraph}{#1} \originalsubparagraph[#1]{#2} \endPDFParSect}%
970 \newcommand{\originalsubparagraphWithOption}[1]%
971   {\PDFParSect{Subparagraph}{#1} \originalsubparagraph{#1} \endPDFParSect}%
```

3.7.5 Minisec

Ein wenig getrennt von den anderen Überschriften ist die im Koma-Script-Paket eingeführt `\minisec`. Sie generiert eine kleine Zwischenüberschrift und wird nicht ins Inhaltsverzeichnis aufgenommen. Sie soll (mittels H) als solche gekennzeichnet werden. Die eigentliche Markierung übernimmt `\everypar`.

minisec Umdefinieren des `\minisec`-Befehls

```
972 \ifthenelse{\boolean{@Access@pdf}}{%
973   \@ifundefined{minisec}{%
974     \let\originalminisec\minisec%
975     \renewcommand{\minisec}{%
976       \@ifstar{\originalminisecWithStar}%
977       {\@ifnextchar[%]
978         {\originalminisecWithTwoOption}%
979     }%
980   }%
981 }{}
```

```

979         {\originalminisecWithOption}%
980     }%
981 }%
982 }%
983 }{}

```

Zuordnung der verschiedenen Aufrufvarianten.

```

984 \newcommand{\originalminisecWithStar}[1]%
985   {\PDFSpezialTextObj{H}\EveryparConfig{H}{false}%
986     \originalminisec*{#1} \endPDFSpezialTextObj}%
987 \newcommand{\originalminisecWithTwoOption}[2]%
988   {\PDFSpezialTextObj{H}\EveryparConfig{H}{false}%
989     \originalminisec{#1}{#2} \endPDFSpezialTextObj}%
990 \newcommand{\originalminisecWithOption}[1]%
991   {\PDFSpezialTextObj{H}\EveryparConfig{H}{false}%
992     \originalminisec{#1} \endPDFSpezialTextObj}%

```

3.8 Blockelemente

Blockelemente sind Strukturen wie Zitatumgebungen. Sie bestehen aus einer besonderen Textumgebung, die spezielle Abschnitte logisch hervorhebt.

3.8.1 Zitatumgebungen

Für Zitatumgebungen steht, in den Standardelementen von PDF, nur das /Quote-Objekt zur Verfügung. Es ist ein spezielles Textobjekt wodurch auch eine Schachtelung von Elementen auf Zeilenebene möglich ist. Den Standardfall ohne weitere Schachtelungen zeigt Abbildung 3.2.

Die Latex-Struktur

```

\begin{quote}
  "Ich bin ein
  kurzes Zitat."
\end{quote}

```

Die PDF-Struktur

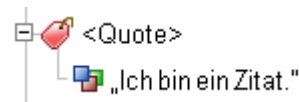


Abbildung 3.2: Struktur einer Zitatumgebung

Das eigentliche Umdefinieren

```

993 \ifthenelse{\boolean{@Access@pdf}}{%

```

quote Umdefinieren der \quote-Umgebung

```

994 \let\originalquote\quote%
995 \let\originalendquote\endquote%
996 \renewenvironment*{quote}%
997   {\PDFSpezialTextObj{Quote}\EveryparConfig{Quote}{false}\originalquote}%
998   {\endPDFMarkContent\originalendquote\endPDFSpezialTextObj}%

```

quotation Umdefinieren der `\quotation`-Umgebung

```
999 %
1000 \let\originalquotation\quotation%
1001 \let\originalendquotation\endquotation%
1002 \renewenvironment*{quotation}%
1003   {\PDFSpezialTextObj{Quote}\EveryparConfig{Quote}{false}\originalquotation}%
1004   {\endPDFMarkContent\originalendquotation\endPDFSpezialTextObj}%
```

verse Umdefinieren der `\verse`-Umgebung

```
1005 %
1006 \let\originalverse\verse%
1007 \let\originalendverse\endverse%
1008 \renewenvironment*{verse}%
1009   {\PDFSpezialTextObj{Quote}\EveryparConfig{Quote}{false}\originalverse}%
1010   {\endPDFMarkContent\originalendverse\endPDFSpezialTextObj}%
1011 }{}
```

3.8.2 Verbatim, Listings und andere

In PDF steht eine `/Code`-Objekt für Computerprogramme und ähnliche Strukturen zur Verfügung. Es soll im folgenden zur Umsetzung der Verbatim-Umgebung herangezogen werden. Bei zukünftigen Umsetzungen von `listings` oder `algorithm` sollte ein ähnliches Vorgehen gewählt werden.

Die Latex-Struktur

```
%begin{verbatim}
  Quelltext%
%end{verbatim}
```

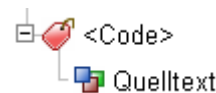
Die PDF-Struktur

Abbildung 3.3: Struktur von Code

verbatim Die folgende Umsetzung funktioniert ohne extra Paket sowie mit den Paketen `verbatim` und `fancyvrb`. Es kommt je verwendeter Verbatim-Umgebung zu einem Fehler („Something’s wrong—perhaps a missing `\item`.“), allerdings hat dieser keine festgestellten Auswirkungen auf das erzeugte Dokument.

```
1012 \ifthenelse{\boolean{@Access@pdf}}{%
1013   \let\originalverbatim\@verbatim%
1014   \renewcommand{\@verbatim}{%
1015     %\PDFStructObj{Div}\empty}%
1016   \PDFSpezialTextObj{Code}
1017   \originalverbatim%
1018 }%
```

```

1019 \let\originalendverbatim\endverbatim%
1020 \renewcommand{\endverbatim}{%
1021   \endPDFMarkContent%
1022   \originalendverbatim%
1023   \endPDFSpezialTextObj%
1024   %\endPDFStructObj%
1025 }%
1026 \expandafter\let\csname endverbatim*\endcsname =\endverbatim%
1027 }{}

```

3.8.3 Theorem

Theoreme dienen der Verwaltung von Definitionen, Merksätzen, Beispielen, Aufgaben... und transportieren damit wichtige logische Informationen die sich in der Struktur widerspiegeln sollten. Da diese Strukturen aber recht flexibel sind, ist kein rechtes Pendant in der PDF-Spezifikation auszumachen. Anbieten tut sich jedoch das abstrakte /Div-Element von dem eigene Strukturen abgeleitet werden könnten. Eine Wiederverwendung des definierten Strukturnames führt jedoch zu Problemen. Zum Einen ist die Sprache der PDF-Objekte bisher Englisch, während der Theoremname praktisch in allen Sprachen definiert sein kann, was zum Anderen auch zu Problemen mit Sonderzeichen (z. B. Umlaute, Akzente...) führt. Daher werden Theoreme vorerst als /Div umgesetzt.

Die Latex-Struktur

```

\begin{definition}
  Ein Theorem ...
\end{definition}

```

Die PDF-Struktur

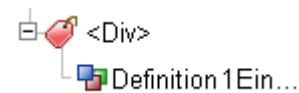


Abbildung 3.4: Struktur eines Theorems

Poteniiell schachtelbar mit z. B. Itemize oder mehrere Absätze.

TODO 2 *vielleicht Argumente auswerten, zur extra Kennzeichnung als heading*

TODO 3 *vielleicht Name in Title übernehmen mit pdfstring*

theorem Umdefinieren der \theorem-Umgebung.

```

1028 \ifthenelse{\boolean{@Access@pdf}}{%

```

überprüfung ob das Paket thmbox geladen ist.

```

1029 \@ifpackageloaded{thmbox}{%
1030   \PackageWarning{accessibility}%
1031     {The thmbox-package isn't yet supported.}%
1032 }{}%

```

Umdefinieren von theroem, wenn das theorem-Paket geladen ist.

```
1033 \@ifpackageloaded{theorem}{%
1034   \newcommand{\@myendtheorem}{%
1035     \@endtheorem%
1036     \endPDFSpezialTextObj%
1037   }%TODO ungetestet
1038   \let\original@thm\@thm%
1039   \gdef\@thm#1#2{%
1040     \PDFSpezialTextObj{Div}%
1041     \EveryparConfig{H}{true}%
1042     \PDFMarkContent%
1043     \global \expandafter \let \csname end#1\endcsname \@myendtheorem%
1044     \original@thm{#1}{#2}%
1045   }%
```

Umdefinieren von theroem ohne das theorem-Paket

```
1046 }{%without theorem-package
1047   \let\original@begintheorem\@begintheorem%
1048   \renewcommand{\@begintheorem}{%
1049     \PDFSpezialTextObj{Div}%
1050     \EveryparConfig{H}{true}%
1051     \PDFMarkContent%
1052     \EveryparConfig{P}{true}%
1053     \original@begintheorem%
1054   }%
1055   \let\original@opargbegintheorem\@opargbegintheorem%
1056   \renewcommand{\@opargbegintheorem}{%
1057     \PDFSpezialTextObj{Div}%
1058     \EveryparConfig{H}{true}%
1059     \PDFMarkContent%
1060     \EveryparConfig{P}{true}%
1061     \original@opargbegintheorem%
1062   }%
1063   \let\original@endtheorem\@endtheorem%
1064   \renewcommand{\@endtheorem}{%
1065     \original@endtheorem%
1066     \endPDFSpezialTextObj%
1067   }%
1068 }%
1069 }%
1070 }{}
```

3.8.4 Aufzählumgebungen

Bei Aufzählungen sieht es im Vergleich zu den Zitatumgebungen schon etwas komplizierter aus. Da in \LaTeX standardmäßig bis zu vier Schachtelungen erlaubt sind.

Wie bei den Zitatumgebungen existiert in PDF laut Spezifikation nur eine Listenstruktur /L. Sie unterliegt einer festen Gliederung (vgl. Abbildung 3.5). Wobei jeder Listeneintrag /LI aus einem optionalen Label /Lb1 und einem obligatorischen Listenkörper /LBody besteht.

Die Latex-Struktur

```
\begin{description}
  \item[Begriff 1] erster Punkt
  \item[Begriff 2] zweiter Punkt
\end{description}
```

Die PDF-Struktur

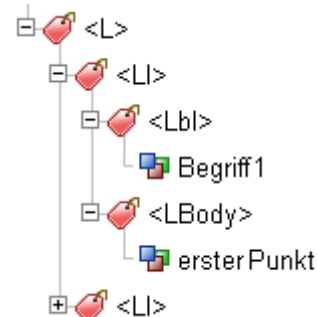


Abbildung 3.5: Struktur einer Liste

Geschachtelte Unterlisten sind auf der Ebene des /LI der übergeordneten einzugliedern.

Variablendeklaration

Im folgenden werden einige Variablen benötigt, um die Elemente zusammzusetzen sowie die Ebenen zu Unterscheiden.

```
1071 \newif\ifItemActive \ItemActivefalse%
1072 \newcounter{ListDepth}%
```

Hilfsmakros

PDFList Dieses Makro initialisiert im einfachsten Fall nach der Beendigung des noch aktiven Textes nur die Liste. D. h. die Variablen werden initialisiert bzw. zurückgesetzt, sollte zuvor schon eine Liste abgearbeitet worden sein. Für den Fall, dass schon einer Liste offen ist, soll in dieser noch das letzte Item abgeschlossen werden. Ausserdem muss der Befehl \item für eine Erkennung umdefiniert werden.

```
1073 \newenvironment{PDFList}{%
1074   \ifItemActive \closeItem\fi%
1075   %Liste beginnen
1076   \addtocounter{ListDepth}{1}%
1077   %\PDFStructObj{L}{\empty}% Sonst Fehler bei Zugriffsprüfung AA
1078   \PDFStructObj{L\arabic{ListDepth}}{\empty}%
1079   %\PDFStructObj{L\romannumeral\theListDepth}{\empty}%
1080 }{%
1081   \ifItemActive \closeItem\fi%
1082   %Liste beenden
1083   \endPDFStructObj%
1084   \addtocounter{ListDepth}{-1}%
1085 }
```

PDFListLabel Diese Umgebung klammert den `\item` Befehl und kennzeichnet somit das Label. Da der `/LBody` in \LaTeX nicht explizit ausgezeichnet ist, wird nach Abschluss des Labels gleich mit dem `/LBody` fortgesetzt.

```
1086 \newenvironment{PDFListLabel}{%
1087   \ifItemActive \closeItem\fi%
1088   \PDFStructObj{LI}{\empty}%
1089   \global\ItemActivetrue%
1090   \PDFSpezialTextObj{Lbl}%
1091   \EveryparConfig{Lbl}{false}%
1092   \PDFMarkContent%
1093 }{%
1094   \endPDFMarkContent%
1095   \endPDFSpezialTextObj%
1096   \PDFSpezialTextObj{LBody}%
1097   \EveryparConfig{LBody}{false}%
1098   %\PDFMarkContent{LBody}% wird über everypar erledigt
1099 }
```

\closeItem Ein zugehöriges Gegenstück, wie bei anderen Befehlen gibt es aufgrund der \LaTeX -Struktur nicht. Somit sollte zu Beginn eines neuen Items oder am Ende der Liste das letzte Item geschlossen werden. Diese Funktionalität kapselt dieses Makro.

```
1100 \newcommand{\closeItem}{% Altes Item abschließen
1101   \endPDFMarkContent%
1102   \endPDFSpezialTextObj{LBody}
1103   \endPDFStructObj%
1104   \global\ItemActivefalse%
1105 }
```

Das eigentliche Umdefinieren

```
1106 \ifthenelse{\boolean{@Access@pdf}}{%
```

itemize Umdefinieren der itemize-Umgebung

```
1107 \let\originalitemize\itemize%
1108 \let\originalenditemize\enditemize%
1109 \renewenvironment{itemize}%
1110   {\begin{PDFList}\originalitemize}%
1111   {%\ifItemActive \closeItem\fi%
1112   \originalenditemize\end{PDFList}}%
1113 %
```

Kennzeichnung der Label für Itemize.

```
1114 \let\originallabelitemi\labelitemi%
1115 \renewcommand{\labelitemi}{%
1116   \begin{PDFListLabel} \originallabelitemi \end{PDFListLabel}}%
1117 \let\originallabelitemii\labelitemii%
1118 \renewcommand{\labelitemii}{%
1119   \begin{PDFListLabel} \originallabelitemii \end{PDFListLabel}}%
1120 \let\originallabelitemiii\labelitemiii%
1121 \renewcommand{\labelitemiii}{%
1122   \begin{PDFListLabel} \originallabelitemiii \end{PDFListLabel}}%
1123 \let\originallabelitemiv\labelitemiv%
1124 \renewcommand{\labelitemiv}{%
1125   \begin{PDFListLabel} \originallabelitemiv \end{PDFListLabel}}%
1126 %
```

enumerate Umdefinieren der enumerate-Umgebung

```
1127 \let\originalenumerate\enumerate%
1128 \let\originalendenumerate\endenumerate%
1129 \renewenvironment{enumerate}%
1130   {\begin{PDFList}\originalenumerate}%
1131   {%\ifItemActive \closeItem\fi%
1132   \originalendenumerate\end{PDFList}}%
1133 %
```

Kennzeichnung der Label für Enumerate.

```
1134 \let\originallabelenumi\labelenumi%
1135 \renewcommand{\labelenumi}{%
1136   \begin{PDFListLabel} \originallabelenumi \end{PDFListLabel}}%
1137 \let\originallabelenumii\labelenumii%
1138 \renewcommand{\labelenumii}{%
1139   \begin{PDFListLabel} \originallabelenumii \end{PDFListLabel}}%
1140 \let\originallabelenumiii\labelenumiii%
1141 \renewcommand{\labelenumiii}{%
1142   \begin{PDFListLabel} \originallabelenumiii \end{PDFListLabel}}%
1143 \let\originallabelenumiv\labelenumiv%
1144 \renewcommand{\labelenumiv}{%
1145   \begin{PDFListLabel} \originallabelenumiv \end{PDFListLabel}}%
1146 %
```

description Umdefinieren der description-Umgebung

```
1147 \let\originaldescription\description%
1148 \let\originalenddescription\enddescription%
1149 \renewenvironment{description}%
1150   {\begin{PDFList}\originaldescription}%
1151   {%\ifItemActive \closeItem\fi%
```

```

1152 \originalenddescription\end{PDFList}}%
1153 %

```

Kennzeichnung der Label für Description.

```

1154 \let\originaldescriptionlabel\descriptionlabel% aus scrept
1155 \renewcommand{\descriptionlabel}[1]{%
1156 \begin{PDFListLabel} \originaldescriptionlabel{#1} \end{PDFListLabel}}%
1157 }{}

```

3.8.5 Formeln

Das PDF-Element /Formula ist für die Auszeichnung von Formeln gedacht (vgl. Abbildung 3.6). Eine logische Differenzierung in eingebettet und freistehende Formeln wird nicht vorgenommen. Dieses Unterscheidungsmerkmal kann durch die unterschiedliche Einbettung in die Struktur wiedergegeben werden. Zum einen kann das Formelobjekt in den Textabsatz eingegliedert werden, zum anderen unter das aktive Section-Objekt. Wie die Struktur für die Formel selbst auszusehen hat zeigt Abbildung 3.6.

Die Latex-Struktur

```

\  

\alt{c^2=a^2+b^2}  

c^{2}=a^{2}+b^{2}  

\}

```

Die PDF-Struktur

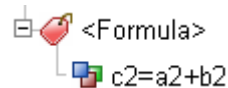


Abbildung 3.6: Struktur einer Formel

Das eigentliche Umdefinieren

TODO 4 alle Formeltypen und Alt-Tag

```

1158 \ifthenelse{\boolean{@Access@pdf}}{%

```

`\[\]` Hier wird die Formelumgebungen, die durch eckige Klammern gekennzeichnet wird ausgezeichnet.

```

1159 \let\originalFormulaBegin\[%
1160 \renewcommand*{\[}{%
1161 \PDFSpezialTextObj{Formula}
1162 \EveryparConfig{Formula}{false}}%
1163 \originalFormulaBegin%
1164 }%
1165 \let\originalFormulaEnd\]%
1166 \renewcommand*{\]}{%
1167 \endPDFMarkContent
1168 \originalFormulaEnd%
1169 \endPDFSpezialTextObj%
1170 }%

```

Die Formelumgebung `\math` greift intern auf `\(\)` zu, ebenso wie `\displaymath` auf `\[\]`, dadurch brauchen diese Umgebungstypen nicht extra behandelt werden.

Um den komplexeren Formelumgebungen wirklich gerecht zu werden, sollten sie eventuell in mehrere Formeln zerlegt und dann in die Struktur eingebunden werden.

equation Im Folgenden wird die equation-Umgebung gekapselt.

```
1171 \let\originalequation\equation%
1172 \let\originalendequation\endequation%
1173 \renewenvironment{equation}%
1174   {\PDFSpezialTextObj{Formula}\EveryparConfig{Formula}{false}\originalequation}%
1175   {\endPDFMarkContent\originalendequation\endPDFSpezialTextObj}%
1176   %
```

eqnarray Auszeichnung des eqnarray, dabei wurde auf eine Umsetzung der Tabelle absichtlich verzichtet, diese dient eher der Darstellung, als der logischen Gliederung.

```
1177 \let\originaleqnarray\eqnarray%
1178 \let\originalendeqnarray\endeqnarray%
1179 \renewenvironment{eqnarray}%
1180   {%\def&\originalamp}% --> das bringt den Fehler inaccessible
1181   \PackageWarning{accessibility}{The 'eqnarray' environment should not be used anymore. It is
1182   \PDFSpezialTextObj{Formula}%
1183   \EveryparConfig{Formula}{false}\originaleqnarray}%
1184   {\endPDFMarkContent\originalendeqnarray\endPDFSpezialTextObj}%
1185 }{}%
```

3.8.6 Gleitumgebungen

Da Gleitumgebungen (Figure, Float) werden von \LaTeX positioniert werden und können möglicherweise auf einer anderen Seite landen. Die zugehörigen Seitenobjekte, die in `/Pg` angegeben werden, sollten bei der Definition dynamisch berechnet werden.

Eine Gleitumgebung (z.B. eine Abbildung, Tabelle oder ein Listing) sollte entsprechend der Abbildung 3.7 umgesetzt werden. Es ist allerdings darauf zu achten, dass `\includegraphics` und ähnliche Befehle auch ohne Gleitumgebung auftauchen können und z. B. in einer `\figure`-Gleitumgebung keinesfalls nur eindeutige Grafikbefehle verwandt werden können. Hier könnten auch einfacher Text oder eine Minipage enthalten sein. Deshalb wird zur Umsetzung eine eigens definiertes `/Float`-Tag verwendet, dass von `/Div` abgeleitet ist. Die geschachtelten Grafiken, Tabellen, Captions werden dieser `/Float`-Struktur untergeordnet. Dies ist die stabilere Lösung, da `\includegraphics` oder `\tabular` auch ohne zugehöriges Gleitobjekt auftreten kann.

Die Latex-Struktur

```
\begin{figure}[htbp]
  \alt{Ich bin das Logo der
    Technischen Universität}
  \includegraphics{/tu_logo}
  \caption{TU-Logo}
\end{figure}
```

Die PDF-Struktur

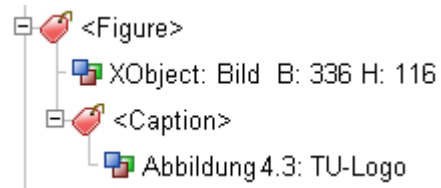


Abbildung 3.7: Struktur einer Grafik

float Umdefinieren der float-Umgebung, diese wird sowohl für die Definition von `\figureend` und `\table` als auch für selbstdefinierte Floatumgebungen verwendet.

```
1186 \ifthenelse{\boolean{@Access@pdf}}{%
1187   \let\original@float\@float%
1188   \let\originalend@float\end@float%
1189   \renewenvironment*{@float}[1]{%
1190     \PDFStructObj{Float}{\csname #1name\endcsname}%
1191     %\global\numberingparsfalse%
1192     \original@float{#1}%
1193   }{%
1194     \originalend@float%
1195     \endPDFMarkContent%
1196     %\global\numberingparstrue%
1197     \endPDFStructObj%
1198   }%
1199 }{}
```

3.8.7 Caption

Eine Bildunterschrift (CM) tritt normalerweise in einer Gleitumgebung auf. Der Befehl kann allerdings auch in einer minipage oder irgendwo anders verwendet werden.

caption Durch das umdefinieren von `\@makecaption` funktioniert diese Umsetzung mit den Standardklassen, den Klassen des KOMA-Scriptes sowie mit dem caption-Paket.

```
1200 \ifthenelse{\boolean{@Access@pdf}}{%
1201   \let\original@@makecaption\@makecaption%
1202   % \renewcommand{\@makecaption}[3]{%
1203   \renewcommand{\@makecaption}[2]{%
1204     \global\numberingparsfalse%
1205     \PDFSpezialTextObj{Caption}%
1206     \EveryparConfig{Caption}{false}%
1207     \PDFMarkContent%
1208     \PackageWarning{accessibility}{begin makecaption}%
1209 %     \original@@makecaption{#1}{#2}{#3}%
1210     \original@@makecaption{#1}{#2}{#3}%
1211     \PackageWarning{accessibility}{end makecaption}%
```

```

1212 \endPDFMarkContent%
1213 \endPDFSpezialTextObj{Caption}%
1214 \global\numberingparstrue%
1215 }%
1216 }{}

```

\captionbelow \captionbeside \captionabove

3.8.8 Tabellen

Eine Tabelle besteht in PDF aus drei großen Teilen, dem Tabellenkopf, dem -körper und dem -fuß. Diese bestehen jeweils aus Tabellenreihe, die wiederum Tabellendatenzellen bzw. Tabellenüberschriftszellen enthalten.

Eine Unterscheidung in Kopf, Körper und Fuß ist in \LaTeX -Tabellen nicht zu finden. Lediglich die Erweiterung longtable bringt ein ähnliches Konzept mit.

Die Latex-Struktur

```

\begin{table}[htbp]
  \begin{tabular}{l|l l}
    \thead{11} & \thead{12} & & \\
    \thead{13} & \\\ \hline
    21 & 22 & 23 & \\
    31 & 32 & 33 & \\
  \end{tabular}
  \caption{meine Tabelle}
\end{table}

```

Die PDF-Struktur

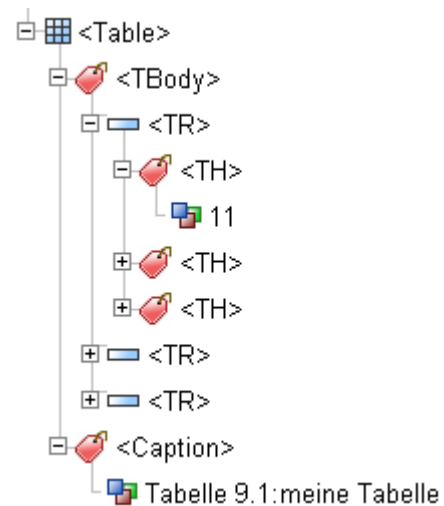


Abbildung 3.8: Struktur einer Tabelle

Variablendeklaration

```

1217 \newif\ifTableHeadCell \global\TableHeadCellfalse%
1218 \newif\ifTableLineActive \global\TableLineActivefalse%
1219 \newif\ifTableCellActive \global\TableCellActivefalse%
1220 \newif\ifAfterKill \global\AfterKillfalse%

```

Hilfsmakro

PDFTable Umschließt die gesamte Tabelle.

```

1221 \newenvironment{PDFTable}{%

```

```

1222 \global\numberingparsfalse%
1223 \PDFStructObj{Table}{\empty}%
1224 \PDFStructObj{TBody}{\empty}%
1225 \global\TableLineActivefalse%
1226 \global\TableCellActivefalse%
1227 }{%
1228 \ifTableLineActive\endPDFTableLine\fi%
1229 \endPDFStructObj{TBody}{\empty}%
1230 \endPDFStructObj{Table}{\empty}%
1231 \global\numberingparstrue%
1232 }%

```

PDFTableLine Eine Tabellenzeile

```

1233 \newenvironment{PDFTableLine}{%
1234 \ifTableCellActive\endPDFTableCell\fi%
1235 \ifTableLineActive\endPDFTableLine\fi%
1236 \global\TableLineActivetrue%
1237 \PDFStructObj{TR}{\empty}%
1238 }{%
1239 \ifTableLineActive%
1240 \endPDFStructObj%
1241 \global\TableLineActivefalse%
1242 \fi%
1243 }%

```

PDFTableCell Eine Tabellenzelle, die Unterscheidung in Überschrifts- und Datenzelle wird vom Autor getroffen. Der zugrunde liegende Wahrheitswert wird in `TableHeadCell` gespeichert.

```

1244 \newenvironment{PDFTableCell}{%
1245 \ifTableCellActive\endPDFTableCell\fi%
1246 \global\TableCellActivetrue%
1247 \PDFSpezialTextObj{TD}%
1248 \EveryparConfig{TD}{false}%
1249 \PDFMarkContent%
1250 }{%
1251 \ifTableCellActive%
1252 \endPDFMarkContent%
1253 \ifTableHeadCell%
1254 \xdef\TextType{TH}%
1255 \global\TableHeadCellfalse%
1256 \fi%
1257 \endPDFSpezialTextObj{TD}%
1258 \global\TableCellActivefalse%
1259 \fi%
1260 }%

```

Das eigentliche Umdefinieren

`tabular` Umdefinieren der `\tabular`-Umgebung.

```
1261 \def\originalamp{&}%
1262 \catcode'\&=\active%
1263 \def&{\originalamp}%
1264
1265 \ifthenelse{\boolean{@Access@pdf}}{%
1266   \let\originaltabular\tabular%
1267   \let\originalendtabular\endtabular%
1268   \renewenvironment*{tabular}{%
1269     \def&{\endPDFTableCell\originalamp\PDFTableCell}%
1270     \PDFTable%
1271     \PDFTableLine%
1272     \PDFTableCell%
1273     %%%%%%%%%%%
1274     \originaltabular%
1275   }{%
1276     %\pdfliteral{EMC}%
1277     \def&{\originalamp}%
1278     \originalendtabular%
1279     %%%%%%%%%%%
1280     \ifTableCellActive\endPDFTableCell\fi%
1281     \ifTableLineActive\endPDFTableLine\fi%
1282     \endPDFTable%
1283   }%
```

Zur Markierung des Tabellenzeilenendes, es ist eine Unterscheidung nötig, je nachdem, ob das Paket `tabularx` geladen ist oder nicht.

```
1284 \@ifpackageloaded{array}{%
1285   \let\originalaryend\@arraycr%
1286   \renewcommand*{\@arraycr}{\endPDFTableCell%
1287     \endPDFTableLine\PDFTableLine\PDFTableCell\originalaryend}%
1288 }{% wenn kein anderes Tabellen-Package
1289   \let\originaltabend\@tabularcr%
1290   \renewcommand*{\@tabularcr}{\endPDFTableCell%
1291     \endPDFTableLine\PDFTableLine\PDFTableCell\originaltabend}%
1292 }%
```

Die Pakete `tabularx` und `longtable` sowie weitere werden bisher nicht behandelt.

```
1293 % \@ifpackageloaded{tabularx}{%
1294 %   \PackageWarning{accessibility}%
1295 %     {The tabularx-package isn't yet fully supported.%
1296 %     You can use the tabular-environemt but not the tabularx.}
1297 % }{%
1298 % \@ifpackageloaded{longtable}{%
1299 %   \PackageWarning{accessibility}%
1300 %     {The longtable-package isn't yet supported.}
```

```

1301 % %\tabularnewline \endhead\endfirsthead\endfoot\endlastfoot
1302 % }{}%
1303 }{}%

```

tabbing Umdefinieren der \tabbing-Umgebung.

```

1304 \ifthenelse{\boolean{@Access@pdf}}{%
1305 \let\originaltabbing\tabbing%
1306 \let\originalendtabbing\endtabbing%
1307 \renewenvironment*{tabbing}{%
1308 \PDFTable%
1309 \let\originalkill\kill%
1310 \renewcommand{\kill}{\global\AfterKilltrue%
1311 \originalkill}%
1312 }%
1313 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1314 \originaltabbing%
1315 }{%
1316 \originalendtabbing%
1317 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1318 \endPDFTable%
1319 }%
1320 \let\original@startfield\@startfield%
1321 \renewcommand{\@startfield}{%
1322 \original@startfield \ifAfterKill\PDFTableCell\fi%
1323 }%
1324 \let\original@stopfield\@stopfield%
1325 \renewcommand{\@stopfield}{%
1326 \ifAfterKill\endPDFTableCell\fi \original@stopfield%
1327 }%
1328 \let\original@startline\@startline%
1329 \renewcommand{\@startline}{%
1330 \ifAfterKill\PDFTableLine\fi \original@startline%
1331 }%
1332 \let\original@stopline\@stopline%
1333 \renewcommand{\@stopline}{%
1334 \original@stopline \ifAfterKill\endPDFTableLine\fi%
1335 }%
1336 }{}

```

3.9 Elemente auf Zeilenebene

3.9.1 Text hervorhebungen

Zeichnet Formatierungen im Fließtext als /Span aus, um sie gesondert hervorzuheben. Eine Auszeichnung von reinen Textdekorationen (z.B. \textbf{ }, \textit{ } ...) ist hierbei jedoch fraglich, da sie auch in Makros verwendet werden und somit möglicherweise mehrfach ausgezeichnet werden, was zum einen zu Problemen in der Struktur führt und zum anderen schnell unübersichtlich wird. Vergleichbare Elemente sind in PDF nicht vorgesehen und auch in XHTML

2.0 soll die Trennung von Inhalt und Layout durch den Wegfall der Elemente (``, `<it>` ...) vollendet werden.

Hingegen transportiert die Struktur `\emph{}` durchaus semantische Informationen. Nämlich das der Text hervorgehoben ist.

Das eigentliche Umdefinieren

emph Die Auszeichnung des `\emph`-Befehls.

```
1337 \ifthenelse{\boolean{@Access@pdf}}{%
1338   \let\originalemph\emph%
1339   \renewcommand{\emph}[1]{%
1340     \begin{PDFInlineObjInText}{Span}%
1341     \originalemph{#1}%
1342     \end{PDFInlineObjInText}%
1343   }%
1344 }
```

3.9.2 Verweise auf andere Textstellen

Für Verweise auf anderen Textstellen bietet PDF die Struktur `/Reference`.

Die Latex-Struktur

```
...S.
\pageref
...
```

Die PDF-Struktur



Abbildung 3.9: Die Struktur einer Referenz

```
1345 \ifthenelse{\boolean{@Access@pdf}}{%
```

Wenn das `hyperref`-Paket geladen ist.

```
1346 \@ifpackageloaded{hyperref}{%
1347   \let\originalsetref\setref%
1348   \renewcommand{\setref}[3]{%
1349     \begin{PDFInlineObjInText}{Reference}%
1350     \originalsetref{#1}{#2}{#3}%
1351     \end{PDFInlineObjInText}}%
1352   %Linkziele%
1353   %\let\originalhyper@anchorstart\hyper@anchorstart%
1354   %\renewcommand{\hyper@anchorstart}%
1355   %{\pdfliteral{/Span <</E (anchorstart)>> BDC EMC}}%
1356   %\originalhyper@anchorstart}%
1357   %\let\originalhyper@anchorend\hyper@anchorend%
1358   %\renewcommand{\hyper@anchorend}{\originalhyper@anchorend
1359   %\pdfliteral{/Span <</E (anchorend)>> BDC EMC}}%
```

```

1360 % Einträge im TOC, LOF, LOT
1361 %\let\originalhyper@linkstart\hyper@linkstart%
1362 %\renewcommand{\hyper@linkstart}{%
1363 % \begin{PDFInlineObjInText}{Reference}%
1364 % \originalhyper@linkstart}%
1365 %\let\originalhyper@linkend\hyper@linkend%
1366 %\renewcommand{\hyper@linkend}{%
1367 % \originalhyper@linkend%
1368 % \end{PDFInlineObjInText}}%
1369 %useacronym --> Kurzform, Glossarseitezahlen,
1370 %Indexseitenzahlen, Glossareinträge, Hyperlink
1371 \let\originalhyperlink\hyperlink%
1372 \renewcommand*{\hyperlink}[2]{%
1373 \ifIndexItemActive\else\begin{PDFInlineObjInText}{Reference}\fi%
1374 %Wenn Index -- folgender Aufruf
1375 % hyperlink{page.\the\toks@}{\the\toks@}%
1376 %Bringt Fehler
1377 \originalhyperlink{#1}{#2}%\relax%
1378 \ifIndexItemActive\else\end{PDFInlineObjInText}\fi%
1379 }%
1380 %href pdfobleme mit pdf 1.3 \@urlbordercolor nicht definiert
1381 \let\originalhyper@linkurl\hyper@linkurl%
1382 \renewcommand{\hyper@linkurl}[2]{%
1383 \begin{PDFInlineObjInText}{Link}%
1384 \originalhyper@linkurl{#1}{#2}%
1385 \end{PDFInlineObjInText}}%
1386 %
1387 \let\originalhyper@linkfile\hyper@linkfile%
1388 \renewcommand{\hyper@linkfile}[3]{%
1389 \begin{PDFInlineObjInText}{Link}%
1390 \originalhyper@linkfile{#1}{#2}{#3}%
1391 \end{PDFInlineObjInText}}%
1392 %Seitenzahlen in Index, anders da anschließend
1393 %keine Texterkennung nötig.
1394 %eigentlich über hyperlink möglich
1395 \let\originalhyperpage\hyperpage%
1396 \renewcommand{\hyperpage}[1]{%
1397 \EveryparConfig{Reference}{true}%
1398 \PDFMarkContent% kein everypar
1399 \originalhyperpage{#1}%
1400 \endPDFMarkContent}%
1401 % URL
1402 \let\originalnolinkurl\nolinkurl%
1403 \renewcommand{\nolinkurl}[1]{%
1404 \begin{PDFInlineObjInText}{Link}%
1405 \originalnolinkurl{#1}%
1406 \end{PDFInlineObjInText}}%

```

Wenn das hyperref-Paket nicht geladen ist.

```

1407 }{% ohne hyperref

```

Umdefinieren des `\ref`-Befehls

```
1408 \let\originalref\ref%
1409 \renewcommand{\ref}[1]{%
1410 \begin{PDFInlineObjInText}{Reference}%
1411 \originalref{#1}%
1412 \end{PDFInlineObjInText}}%
1413 %
```

Umdefinieren des `\pageref`-Befehls

```
1414 \let\originalpageref\pageref%
1415 \renewcommand{\pageref}[1]{%
1416 \begin{PDFInlineObjInText}{Reference}%
1417 \originalpageref{#1}%
1418 \end{PDFInlineObjInText}}%
1419 }%
1420 }{}
```

Diese Umsetzung funktioniert auch mit dem `varioref`-Paket, da dieses intern auf die Definitionen von `\ref` bzw. `\pageref`. Die korrekte Auszeichnung sowie die Einbindung der Referenzen funktioniert auch wenn das `hyperref`-Paket geladen ist.

cite Umdefinieren des `\cite`-Befehls, der auf das Literaturverzeichnis verweist.

```
1421 \ifthenelse{\boolean{@Access@pdf}}{%
1422 \let\originalcite\cite%
1423 \renewcommand{\cite}[2][__empty__]{% #1 Name des Eintages
1424 \begin{PDFInlineObjInText}{Reference}%
1425 \ifthenelse{\equal{#1}{__empty__}}%
1426 {\originalcite{#2}}%
1427 {\originalcite[#1]{#2}}%
1428 \end{PDFInlineObjInText}}%
1429 }%
1430 }{}
```

Eine getrennte Auszeichnung der Glossareninträge ist nicht mehr nötig. Das `glossary` greift auf `\hyperlink` zurück. Auch möglich Seitenbezüge im Glossar werden über `\hyperlink` aktiviert.

3.9.3 eingebettete Objekte im Textfluss

\verb An dieser Stelle erfolgt das Umdefinieren der eingebetteten Codeumgebung, die durch `\verb` gekennzeichnet wird.

```
1431 \ifthenelse{\boolean{@Access@pdf}}{%
1432 \let\originalverb\verb%
1433 \renewcommand{\verb}{%
1434 \begin{PDFInlineObjInText}{Code}%
1435 \originalverb%
1436 }%
1437 \let\originalverb@egroup\verb@egroup%
```

```

1438 \renewcommand{\verb@egroup}{%
1439 \originalverb@egroup%
1440 \end{PDFInlineObjInText}%
1441 }%
1442 }{}

```

`\(\)` An dieser Stelle erfolgt das Undefinieren der eingebetteten Formelumgebungen, die durch runde Klammern gekennzeichnet wird.

```

1443 \let\originalFormulaTextBegin\(%
1444 \renewcommand*{\}{%
1445 \PDFInlineObjInText{Formula}%
1446 \originalFormulaTextBegin%
1447 }%
1448 \let\originalFormulaTextEnd\)%
1449 \renewcommand*{\)}{%
1450 \originalFormulaTextEnd%
1451 \endPDFInlineObjInText%
1452 }%

```

3.9.4 Fußnoten

Eine Fußnote besteht generell aus zwei Bestandteilen, der Markierung im Text (footnotemark) und der eigentlichen Fußnote am Seitenende (footnotetext). Beide Teile müssen sinnvoll in die Struktur eingegliedert werden. Hierzu wird die Lesereihenfolge der Elemente im Strukturbaum geändert, sodass der Text an Ort und Stelle verfügbar ist und nicht erst am Seitenende (nach „zig“ Absätzen) vorgelesen wird (vgl. Abbildung 3.10).

Die Latex-Struktur

```

...Fußnote
\footnote{Fußnotentext}
...

```

Die PDF-Struktur

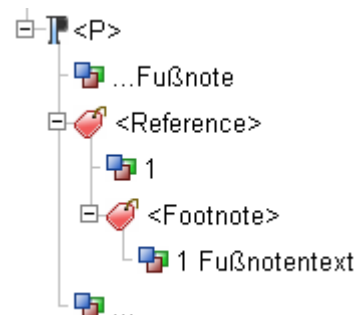


Abbildung 3.10: Fußnotenstruktur im Absatz

TODO 5 Fußnoten außerhalb von Text sind im Moment nicht vorgesehen. → Flexibilisierung der Schachtelung. Also z.B. in Tabelle, überschrift ...

Variablendeklaration

```
1453 \newcounter{PDFFootnotemark}%
1454 \newcounter{PDFFootnotetext}%
1455 \newcounter{ObjNum}
```

Hilfsmakros

PDFFootnote umschließt die gesamte Fußnotenstruktur.

```
1456 \newenvironment{PDFFootnote}{%
1457   \global\numberingparsfalse%
1458   \pdfobj reserveobjnum%
1459   \setcounter{PDFFootnotemark}{\pdflastobj}%
1460   \pdfobj reserveobjnum%
1461   \setcounter{PDFFootnotetext}{\pdflastobj}%
1462 }{%
1463   %\EveryparConfig{\lastEveryparType}{\HelpBool}%
1464   \global\numberingparstrue%
1465   \EveryparConfig{\lastEveryparType}{false}%
1466   \PDFMarkContent%
1467 }
```

PDFFootnoteReference Die eigentliche Referenz auf die Fußnote im Text. Sie setzt sich aus dem markierten Inhalt (MCID) und der Fußnote am Seitenende zusammen.

```
1468 \newenvironment{PDFFootnoteReference}{%
1469   \xdef\HelpBool{\InlineObj}%
1470   \EveryparConfig{Reference}{obj}%
1471   \setcounter{ObjNum}{\theTaggedObj}%
1472   \PDFMarkContent%
1473 }{%
1474   \endPDFMarkContent%
1475   \writeComplexTextObj{\thePDFFootnotemark}%
1476     {\theObjNum \space \thePDFFootnotetext \space 0 R}%
1477     {/Reference}{\theTextObjNum}{Page}%
1478   \xdef\TextArray{\TextArray \theObjHelp\space 0 R \space}%
1479 }
```

PDFFootnoteText Die eigentliche Fußnote am Seitenende. Sie wird als Kind der Fußnotenreferenz in den Strukturbaum eingefügt.

```
1480 \newenvironment{PDFFootnoteText}{%
1481   \EveryparConfig{Note}{obj}%
1482   \setcounter{ObjNum}{\theTaggedObj}%
1483   \PDFMarkContent%
1484 }{%
1485   \endPDFMarkContent%
1486   \writeComplexTextObj%
1487     {\thePDFFootnotetext}{\theObjNum}%

```



```
1488     {\Footnote}{\thePDFFootnotemark}{Page}%
1489 }
```

Das eigentliche Umdefinieren

Die Befehle stammen aus der source2e-Dokumentation.

```
1490 \ifthenelse{\boolean{Access@pdf}}{%
```

Umdefinieren der `\footnotemark`

```
1491 \let\original@footnotemark\@footnotemark%
1492 %Fußnotenreferenz im Text
1493 \renewcommand{\@footnotemark}{%
1494     \begin{PDFFootnoteReference}%
1495     \original@footnotemark%
1496     \end{PDFFootnoteReference}%
1497 }%
```

Umdefinieren der `\footnotetext`

```
1498 \let\original@makefntext\@makefntext%
1499 %Fußnotentext am Seitenende
1500 \renewcommand{\@makefntext}[1]{%
1501     \begin{PDFFootnoteText}%
1502     \original@makefntext{#1}%
1503     \end{PDFFootnoteText}%
1504 }%
```

Umdefinieren der gesamten Fußnote `\footnote`

```
1505 \let\originalfootnote\footnote%
1506 \def\footnote{\@ifnextchar[{\@xxfootnote}{\@xfootnote}}%
1507 \def\@xxfootnote[#1]#2{%
1508     \begin{PDFFootnote}%
1509     \originalfootnote[#1]{#2}%
1510     \end{PDFFootnote}%
1511 }%
1512 \def\@xfootnote#1{%
1513     \begin{PDFFootnote}%
1514     \originalfootnote{#1}%
1515     \end{PDFFootnote}%
1516 }%
1517 }{}
```

3.10 Verzeichnisse

Zahlreiche Verzeichnisse stehen in \LaTeX zur Verfügung. Ihre logische Auszeichnung kann Nutzern assistiver Technologien den Zugang zum Dokument erleichtern.

3.10.1 Inhaltsverzeichnis und die Listen der Float-Objekte

Die Latex-Struktur

```
...
\tableofcontents
  \contentsline {chapter}%
    {Abbildungsverzeichnis}%
    {3}{chapter*.2}
...
```

Die PDF-Struktur

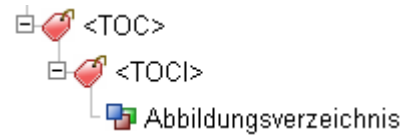


Abbildung 3.11: Struktur eines Inhaltsverzeichnisses

Das eigentliche Umdefinieren

```
1518 \ifthenelse{\boolean{@Access@pdf}}{%
1519   \let\original@starttoc@starttoc%
1520   \renewcommand{\@starttoc}[1]{%
1521     \ifthenelse{\equal{#1}{toc}}{% Table of content
1522       \PDFSpezialTextObj{TOC}\EveryparConfig{TOCI}{true}%
1523     }{}%
1524     \ifthenelse{\equal{#1}{lot}}{% List of Tables
1525       \PDFSpezialTextObj{TOT}\EveryparConfig{TOTI}{true}%
1526     }{}%
1527     \ifthenelse{\equal{#1}{lof}}{% List of figures
1528       \PDFSpezialTextObj{TOF}\EveryparConfig{TOFI}{true}%
1529     }{}%
1530     %\ifthenelse{\equal{#1}{brf}}{}{}% Bibliography
1531     \original@starttoc{#1}%
1532     \ifthenelse{\equal{#1}{toc} \or \equal{#1}{lot} \or \equal{#1}{lof}}{%
1533       \endPDFMarkContent%
1534       \endPDFSpezialTextObj%
1535     }{}%
1536   }%
1537 }{}
```

Verschieben des `\endPDFMarkContent`, damit wird es am Ende der letzten Seite und nicht erst oben auf der neuen ausgeführt.

```
1538 \ifthenelse{\boolean{@Access@pdf}}{%
1539   \let\originalcontentsline\contentsline
1540   \@ifpackageloaded{hyperref}{%then: Mit hyperref
1541     \renewcommand{\contentsline}[4]{%
1542       \originalcontentsline{#1}{#2}{#3\protect\endPDFMarkContent}{#4}%
1543     }%
1544   }{%else: ohne Hyperref
1545     \renewcommand{\contentsline}[3]{%
1546       \originalcontentsline{#1}{#2}{#3\protect\endPDFMarkContent}%
1547     }%
1548   }%
1549 }{}
```

3.10.2 Literaturverzeichnis

Das Literaturverzeichnis (Bibliography) besteht aus einzelnen Literaturverzeichniseinträgen (BibEntry), die im Fließtext mit Literaturverweisen referenziert werden können.

Die Latex-Struktur

```
\begin{thebibliography}{AFF99}
  \bibitem[AFF99]{ansorge:1999}...
\end{thebibliography}
```

Die PDF-Struktur



Abbildung 3.12: Struktur des Literaturverzeichnisses

Variablendeklaration

```
1550 \newif\ifBibItemActive \BibItemActivefalse%
```

Das eigentliche Umdefinieren

Die gewählte Variante funktioniert sowohl mit als auch ohne BibTeX.

Umdefinieren der umschließenden \thebibliography-Umgebung.

```
1551 \ifthenelse{\boolean{@Access@pdf}}{%
1552   \let\originalthebibliography\thebibliography%
1553   \let\originalendthebibliography\endthebibliography%
1554   \renewenvironment{thebibliography}{%
1555     \originalthebibliography%
1556     %\PDFStructObj{Bibliography}% geht hier nicht in bibitem realisiert
1557   }{%
1558     \originalendthebibliography%
1559     \endPDFMarkContent%
1560     \endPDFSpezialTextObj{\LBody}
1561     \endPDFStructObj{\BibItem}
1562     \global\BibItemActivefalse%
1563     \endPDFStructObj{Bibliography}
1564   }%
1565 }
```

Umdefinieren des \bibitem-Befehls.

```
1566 \ifthenelse{\boolean{@Access@pdf}}{%
1567   \let\originalbibitem\bibitem%
1568   \renewcommand{\bibitem}[2][__empty__]{% #1 [Label] #2 Eintrag
1569     \ifBibItemActive% schon welche
1570     \endPDFMarkContent%
1571     \endPDFSpezialTextObj{\LBody}
1572     \endPDFStructObj{\BibItem}
```

```

1573     \global\BibItemActivefalse%
1574     \else% erstes Item
1575     \PDFStructObj{Bibliography}{\empty}%
1576     \fi%
1577     \global\BibItemActivetrue%
1578     \PDFStructObj{BibItem}{\empty}%
1579     \PDFSpezialTextObj{Lb1}%
1580     \EveryparConfig{Lb1}{false}%
1581     \PDFMarkContent%
1582     \ifthenelse{\equal{#1}{__empty__}}%
1583       {\originalbibitem{#2}}%
1584       {\originalbibitem[#1]{#2}}%
1585     %\endPDFMarkContent% Zu früh, Text wird erst mit everypar gestetzt
1586     \endPDFSpezialTextObj%
1587     \PDFSpezialTextObj{LBody}%
1588     \EveryparConfig{LBody}{false}%
1589     %\PDFMarkContent{LBody}% wird über everypar erledigt
1590   }%
1591 }{}

```

3.10.3 Index

Das Stichwortverzeichnis geht häufig über mehrere Spalten und Seiten.

TODO 6 Dabei ist der Umbruch unbedingt zu beachten. → Was passiert derzeit?

Die Latex-Struktur

```

\begin{theindex}
  \item B"achlein, 17
\end{theindex}

```

Die PDF-Struktur



Abbildung 3.13: Struktur des Index

Variablendeklaration

```

1592 \newif\ifIndexItemActive \IndexItemActivefalse%

```

Das eigentliche Umdefinieren

Umdefinieren der umschließenden `\theindex`-Umgebung.

TODO 7 Nur wenn das Paket `index` geladen ist.

```

1593 \ifthenelse{\boolean{@Access@pdf}}{%
1594   \let\originaltheindex\theindex%
1595   \let\originalendtheindex\endtheindex%
1596   \renewenvironment{theindex}{%
1597     \expandafter\originaltheindex\relax%
1598   }{%
1599     \endPDFMarkContent%
1600     \originalendtheindex%
1601     \ifIndexItemActive%
1602       \endPDFSpezialTextObj%
1603       \global\IndexItemActivefalse%
1604     \fi
1605     \endPDFStructObj%{Index}%
1606   }%
1607 }{}

```

Umdefinieren des `\@idxitem`-Befehls.

```

1608 \ifthenelse{\boolean{@Access@pdf}}{%
1609   \let\original@idxitem\@idxitem%
1610   \renewcommand*\@idxitem{%
1611     \ifIndexItemActive% schon welche
1612       \endPDFMarkContent%
1613       \endPDFSpezialTextObj%
1614       \global\IndexItemActivefalse%
1615     \else% erstes Item
1616       \PDFStructObj{Index}%
1617     \fi%
1618     \global\IndexItemActivetrue%
1619     \PDFSpezialTextObj{IndexEntry}%
1620     \EveryparConfig{IndexEntry}{false}%
1621     \original@idxitem%
1622   }%
1623 }{}

```

TODO 8 *subitem und subsubitem getrennt behandeln um die Schachtelung zu erhalten.*

3.11 Layoutbefehle

Befehle, die ausschließlich dem Layout dienen, werden nicht in den Strukturbaum übernommen. Hier ist stattdessen eine Auszeichnung als /Artefakt vorgesehen.

3.11.1 Kopf- und Fußzeilen als Artefakte

Kopf- und Fußzeilen zählen zu den Artefakten, die sich aus der Seitenaufteilung ergeben. Sie sind folglich als solche (`/Type /Page`) zu kennzeichnen.

Hilfsmakro

PDFPageArtefakt Umschließende Struktur für ein Artefakt der Seitenaufteilung.

```
1624 \newenvironment*{PDFPageArtefakt}{%
1625   \pdfliteral{/Artifact <</Type /Pageination>> BDC}%
1626 }{%
1627   \pdfliteral{EMC}%
1628 }
```

Das eigentliche Umdefinieren

Da Scrpge optimal mit den Klassen des Koma-Scripts zusammenarbeitet, funktioniert es mit scrpge2.

TODO 9 Funktionstüchtigkeit mit fancyheader und Standardklassen

```
1629 \ifthenelse{\boolean{@Access@pdf}}{%
1630   \let\original@thehead\@thehead%
1631   \renewcommand*{\@thehead}{%
1632     \ifthenelse{\equal{\original@thehead}{\empty}}{ }{%
1633       \begin{PDFPageArtefakt}%
1634       \original@thehead%
1635       \end{PDFPageArtefakt}%
1636     }%
1637   }%
1638   \let\original@thefoot\@thefoot%
1639   \renewcommand*{\@thefoot}{%
1640     \ifthenelse{\equal{\original@thefoot}{\empty}}{ }{%
1641       \begin{PDFPageArtefakt}%
1642       \original@thefoot%
1643       \end{PDFPageArtefakt}%
1644     }%
1645   }%
1646 }
```

3.11.2 Linien als Artefakte

Linien und andere dekorative Inhalte sind laut PDF-Spezifikation als /Artefakte auszuzeichnen. Normale Linien werden in Screenreadern nicht vorgelesen. Speziell die automatische Füllstruktur (`\dotfill`) wird aber durch ASCII-Zeichen gesetzt, d.h. sie wird im Screenreader als „Punkt Punkt ...“ vorgelesen. Dies stört den Lesefluss erheblich.

Hilfsmakros

PDFLayoutArtefakt Umschließende Struktur für ein Layout-Artefakt.

TODO 10 Kennzeichnung als Artefakt vom Typ /Layout, dazu sollten weitere Parameter (wie die BoundingBox) in angegebene werden, damit zukünftig das Reflow adäquat funktionieren kann.

```
1647 \newenvironment*{PDFLayoutArtefakt}{%
1648   \numberingparsfalse%
1649   \pdfliteral{/Artifact <</Type /Layout>> BDC}%
1650 }{%
1651   \pdfliteral{EMC}%
1652   \numberingparstrue%
1653 }
```

Das eigentliche Umdefinieren

Anpassen des \dotfill-Befehls.

```
1654 \ifthenelse{\boolean{@Access@pdf}}{%
1655   \let\originaldotfill\dotfill%
1656   \renewcommand*{\dotfill}{%
1657     \begin{PDFLayoutArtefakt}%
1658     \originaldotfill%
1659     \end{PDFLayoutArtefakt}%
1660   }%
```

Anpassen des \footnoterule-Befehls. Dieser greift auf hrule zurück und bereite Probleme beim generellen Umdefinieren.

```
1661 \let\originalfootnoterule\footnoterule%
1662 \renewcommand*\footnoterule{%
1663   \let\hrule\originalhrule%
1664   \begin{PDFLayoutArtefakt}%
1665   \originalfootnoterule%
1666   \end{PDFLayoutArtefakt}%
1667   \let\originalhrule\hrule%
1668 }
```

Anpassen des \hrule-Befehls.

```
1669 %\vrule height1ex depth0pt width1ex
1670 %\hrule height1ex depth0pt width1ex
1671 %
1672 %hrulefill, hline cline, toprule, midrule, bottomrule, cmidrule? greifen auf hrule zu
1673 %Klappt nicht immer mit Argumentübergabe
1674 \let\originalhrule\hrule%
1675 \def\hrule#1#2{%
1676   \ifthenelse{\equal{#2}{\z@}}{\}\{\begin{PDFLayoutArtefakt}}%
```

```

1677 \originalhrule#1#2%
1678 \ifthenelse{\equal{#2}{\z@}}{\}\{\end{PDFLayoutArtefakt}}%
1679 }%

```

Ebenso sollten sämtliche Tabellenrahmen, Linien in Kopf- und Fußzeile oder Die Linie vor den Fußnoten markiert werden. Am sinnvollsten erscheint die Umdeklaration der `\hrule` und `\vrule` Anweisung. Auf diese wird in den meisten Fällen zurückgegriffen.

```

1680 %vline (2), @arrayrule(2?) greift auf vrule zu
1681 %Klappt nicht mit Argumentübergabe
1682 %\let\originalvrule\vrule%
1683 %\def\vrule#1#2{%
1684 % \begin{PDFLayoutArtefakt}%
1685 % \originalvrule#1#2%
1686 % \end{PDFLayoutArtefakt}%
1687 % }%
1688 }{}

```

Gepunktete Linien, wie sie im Inhaltsverzeichnis mittels `\dottedtocline` erzeugt werden, werden auch als solches (nämlich „Punkt Punkt ...“) vorgelesen. Hierzu wurde die Originaldefinition aus `source2e [BCJ+00]` um die `pdfliterale` ergänzt, wodurch die Linie als Artefakt gekennzeichnet ist und nicht vorgelesen wird.

```

1689 \ifthenelse{\boolean{@Access@pdf}}{\%
1690 \def@dottedtocline#1#2#3#4#5{%
1691 \ifnum #1>\c@tocdepth \else%
1692 \vskip \z@ \@plus.2\p@%
1693 {\leftskip #2\relax \rightskip \@tocrmarg %
1694 \parfillskip -\rightskip%
1695 \parindent #2\relax\@afterindenttrue%
1696 \interlinepenalty\@M%
1697 \leavevmode%
1698 \@tempdima #3\relax%
1699 \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip%
1700 {#4}\nobreak%
1701 \begin{PDFLayoutArtefakt}%
1702 \leaders\hbox{${\m@th \mkern %
1703 \dotsep mu\hbox{.}\mkern \dotsep mu$}\hfill%
1704 \end{PDFLayoutArtefakt}%
1705 \nobreak%
1706 \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
1707 \par}%
1708 \fi%
1709 }%
1710 }{}

```

3.11.3 Titelseite

Die Titelseite ist sehr von der Gestaltungsfreiheit der Autoren geprägt. Die Standardelemente `\title{}`, `\author{}` und weitere werden oft zu layouttechnischen Zwecken verwandt, so dass

eine inhaltliche Auszeichnung in den Augen der Autorin wenig Sinn macht. Damit die Strukturen, die im Bereich des Titels auftauchen einen sinnvollen Rahmen bekommen, wird der durch `\maketitle` erzeugte Inhalt in die Struktur `/Sect` geschachtelt.

```
1711 \ifthenelse{\boolean{@Access@pdf}}{%
1712   \let\originalmaketitle\maketitle%
1713   \renewcommand{\maketitle}{%
1714     \PDFStructObj{Div}{Titlepage}%
1715     \EveryparConfig{P}{false}%
1716     %
1717     \originalmaketitle%
1718     \endPDFMarkContent%
1719     \endPDFStructObj%
1720   }%
1721 }{}%
1722
```

3.12 Verträglichkeit mit anderen Dokumentklassen

3.13 Verträglichkeit mit anderen Paketen

3.13.1 Das multicolumn-Paket

Wird wie alle anderen Umgebungen unterstützt. Solange sich die gesamte Umgebung auf einer Seite befindet funktioniert alles, wie gehabt. Dass Seitenumbrüche noch nicht zuverlässig erkannt werden können, treten auch hier mögliche Probleme auf. Eine Verwendung sollte nur mit anschließender Überprüfung des Ergebnisdokumentes erfolgen.

Die Befehle `\twocolumn` und `\onecolumn` aus PLAIN \TeX funktionieren mit den gleichen Einschränkungen.

3.13.2 Das graphics-Paket

TODO 11 Die anderen Befehle des *graphicx*-Paketes. (*wrapfigure...*)

```
1723 \ifthenelse{\boolean{@Access@pdf}}{%
1724   \@ifpackageloaded{graphicx}{%
1725     \let\originalincludegraphics\includegraphics%
1726     \renewcommand{\includegraphics}[2][__empty__]{%
1727       \global\numberingparsfalse%
1728       % \PDFInlineObjInText{Figure}%
1729       \PDFSpezialTextObj{Figure}%
1730       \EveryparConfig{Figure}{false}%
1731       \PDFMarkContent%
1732       \ifthenelse{\equal{#1}{__empty__}}%
1733         {\originalincludegraphics{#2}}%
1734         {\originalincludegraphics[#1]{#2}}%
1735       % \endPDFInlineObjInText%

```

```

1736 \endPDFMarkContent%
1737 \endPDFSpezialTextObj{Figure}%
1738 \global\numberingparstrue%
1739 }%
1740 }{}%
1741 }{}

```

3.13.3 Das picture-Paket

Da das picture die Picture-Umgebung transparent umdefiniert, funktioniert die Auszeichnung sowohl wenn das Paket geladen ist. Auch die Erweiterungen trees zum Zeichnen von binären und tertiären Bäumen, bar zum Erstellen vom Balkendiagrammen sowie curves zum Zeichnen beliebiger Kurven kann verwendet werden .

```

1742 \ifthenelse{\boolean{@Access@pdf}}{%
1743 \let\originalpicture\picture%
1744 \let\originalendpicture\endpicture%
1745 \renewenvironment{picture}{%
1746 \global\numberingparsfalse%
1747 \PDFSpezialTextObj{Figure}%
1748 \EveryparConfig{Figure}{false}%
1749 \PDFMarkContent%
1750 \originalpicture%
1751 }{%
1752 \originalendpicture%
1753 \endPDFMarkContent%
1754 \endPDFSpezialTextObj{Figure}%
1755 \global\numberingparstrue%
1756 }%
1757 }{}

```

3.13.4 Das babel-Paket

`\convertLanguageInCode` Dieses Makro konvertiert den übergebenen Sprachstring {#1} in den PDF bekannten Zwei-Buchstaben-Kode. Das Ergebnis wird in der Variablen LanguageCode gespeichert.

```

1758 \newcommand{\convertLanguageInCode}[1]{%
1759 \gdef\LanguageCode{}%
1760 \ifthenelse{\equal{#1}{\string danish}}{\gdef\LanguageCode{/Lang(DA)}}{}%
1761 \ifthenelse{\equal{#1}{\string german}}{\gdef\LanguageCode{/Lang(DE)}}{}%
1762 \ifthenelse{\equal{#1}{\string ngerman}}{\gdef\LanguageCode{/Lang(DE)}}{}%
1763 \ifthenelse{\equal{#1}{\string germanb}}{\gdef\LanguageCode{/Lang(DE)}}{}%
1764 \ifthenelse{\equal{#1}{\string austrian}}{\gdef\LanguageCode{/Lang(DE)}}{}%
1765 \ifthenelse{\equal{#1}{\string naustrian}}{\gdef\LanguageCode{/Lang(DE)}}{}%
1766 \ifthenelse{\equal{#1}{\string english}}{\gdef\LanguageCode{/Lang(EN)}}{}%
1767 \ifthenelse{\equal{#1}{\string USenglish}}{\gdef\LanguageCode{/Lang(EN-US)}}{}%
1768 \ifthenelse{\equal{#1}{\string american}}{\gdef\LanguageCode{/Lang(EN-US)}}{}%
1769 \ifthenelse{\equal{#1}{\string UKenglish}}{\gdef\LanguageCode{/Lang(EN-GB)}}{}%
1770 \ifthenelse{\equal{#1}{\string british}}{\gdef\LanguageCode{/Lang(EN-GB)}}{}%

```

```

1771 \ifthenelse{equal{#1}{string canadian}}{\gdef\LanguageCode{/Lang(EN)}}{}%
1772 \ifthenelse{equal{#1}{string australian}}{\gdef\LanguageCode{/Lang(EN)}}{}%
1773 \ifthenelse{equal{#1}{string newzealand}}{\gdef\LanguageCode{/Lang(EN)}}{}%
1774 \ifthenelse{equal{#1}{string finnish}}{\gdef\LanguageCode{/Lang(FI)}}{}%
1775 \ifthenelse{equal{#1}{string french}}{\gdef\LanguageCode{/Lang(FR)}}{}%
1776 \ifthenelse{equal{#1}{string francais}}{\gdef\LanguageCode{/Lang(FR)}}{}%
1777 \ifthenelse{equal{#1}{string canadien}}{\gdef\LanguageCode{/Lang(FR)}}{}%
1778 \ifthenelse{equal{#1}{string acadian}}{\gdef\LanguageCode{/Lang(FR)}}{}%
1779 \ifthenelse{equal{#1}{string italian}}{\gdef\LanguageCode{/Lang(IT)}}{}%
1780 \ifthenelse{equal{#1}{string norsk}}{\gdef\LanguageCode{/Lang(NO)}}{}%
1781 \ifthenelse{equal{#1}{string nynorsk}}{\gdef\LanguageCode{/Lang(NO)}}{}%
1782 \ifthenelse{equal{#1}{string portugues}}{\gdef\LanguageCode{/Lang(PT)}}{}%
1783 \ifthenelse{equal{#1}{string portuguese}}{\gdef\LanguageCode{/Lang(PT)}}{}%
1784 \ifthenelse{equal{#1}{string brazilian}}{\gdef\LanguageCode{/Lang(PT-BR)}}{}%
1785 \ifthenelse{equal{#1}{string brazil}}{\gdef\LanguageCode{/Lang(PT-BR)}}{}%
1786 \ifthenelse{equal{#1}{string swedish}}{\gdef\LanguageCode{/Lang(SV)}}{}%
1787 \ifthenelse{equal{#1}{string spanish}}{\gdef\LanguageCode{/Lang(ES)}}{}%
1788 % not surreported in babel:
1789 % Chinese (/Lang{ZH})
1790 % Korean (/Lang{KO}).
1791 \ifthenelse{equal{\LanguageCode}{}}{}%
1792 % comparing \language is tricky. See babel package documentation for more information
1793 \PackageWarning{accessibility}{The chosen language (#1) is not supported %
1794 by Adobe Reader 6.0.}%
1795 }{}%
1796 }

```

Auszeichnung der Dokumentenhauptsprache

Am Anfang des eigentlichen Dokumentes wird dann die Hauptsprache des PDF-Dokumentes bestimmt und gesetzt. Zusätzlich wird die aktuelle Sprache initialisiert um bei späteren Änderungen wirkliche von Dopplungen zu unterscheiden.

TODO 12 *Nur wenn babel geladen wurde.*

```

1797 \ifthenelse{\boolean{@Access@pdf}}{}%
1798 \AtBeginDocument{%
1799   \gdef\DocumentLanguage{\language}%
1800   \gdef\ActualLanguage{\language}%
1801   \convertLanguageInCode{\language}%
1802   \pdfcatalog{% Catalog dictionary of PDF output.
1803     \LanguageCode% Setzt die Sprache
1804   }%
1805 }%
1806 }{}

```

Auszeichnung von Sprachwechseln

Hilfsmakro

```
1807 \newcommand{\recognizeLanguageChange}[1]{%
1808   \ifthenelse{\equal{#1}{\ActualLanguage}}{%
1809     %keine änderung zu vorher
1810   }{%
1811     \gdef\ActualLanguage{#1}%
1812     \convertLanguageInCode{\languagename}}%
1813   \ifthenelse{\equal{#1}{\DocumentLanguage}}{%
1814     \global\LanguageDifffalse%
1815   }{%
1816     \global\LanguageDifftrue%
1817   }%
1818 }
```

`\selectlanguage` `\selectlanguage{Sprache}` vollständige Ersetzung bis zum Dokumentenende oder der nächsten änderung. Wenn die neu aktivierte Sprache von der vorherigen abweicht, wird `LanguageDiff` war und alle nun erzeugen Objekte bekommen ein passendes Sprachattribut.

```
1819 \ifthenelse{\boolean{@Access@pdf}}{%
1820   \@ifpackageloaded{babel}{%
1821     \let\originalselectlanguage\selectlanguage%
1822     \renewcommand{\selectlanguage}[1]{%
1823       \originalselectlanguage{#1}%
1824       \recognizeLanguageChange{#1}%
1825     }%
1826   }
```

`otherlanguage` Da die Umgebung `otherlanguage` beliebige Befehle enthalten kann, scheint der Autorin eine umschließende Umgebung fehleranfällig, es könnte so unsinnigen Verschachtelungen kommen. So dass hier das gleiche Vorgehen wie bei `\selectlanguage` gewählt wurde.

TODO 13 `\begin{otherlanguage}{Sprache}` lokale änderung auch in Sternform

TODO 14 Am Anfang der Umgebung doppelte Abfrage durch die Wiederverwendung von `selectlanguage`? sollte eventuell beseitigt werden.

```
1826   \let\originalotherlanguage\otherlanguage%
1827   \let\originalendotherlanguage\otherlanguage%
1828   \long\def\otherlanguage#1{%
1829     \csname selectlanguage \endcsname{#1}%
1830     \ignorespaces%
1831     \recognizeLanguageChange{#1}%
1832   }%
```

```

1833     \long\def\endotherlanguage{%
1834     \originalTeX%
1835     \global\@ignoretrue\ignorespaces%
1836     \recognizeLanguageChange{\language}%
1837     }%

```

foreignlanguage Der Befehl `\foreignlanguageSpracheInhalte` ändert die Sprache nur für kleine Textbereiche, bei denen die Sprachänderung mittels `/Span` in den `ContentStream` eingefügt wird. Eine Einordnung in den Strukturbaum kann laut [Ado04] entfallen.

```

1838     \let\originalforeignlanguage\foreignlanguage%
1839     \renewcommand{\foreignlanguage}[2]{%
1840     \convertLanguageInCode{\string #1}%
1841     \pdfliteral{/Span <<\LanguageCode>> BDC}%
1842     \originalforeignlanguage{#1}{#2}%
1843     \pdfliteral{EMC}%
1844     \convertLanguageInCode{\language}%
1845     }%
1846     }{}%
1847     }{}

```

3.13.5 Das `makeidx`-Paket

3.13.6 Das `glossary`-Paket

Glossar

Die Optionen `altlist` und `list` des `glossary`-Paketes schreiben die Glossareinträge als Definitionsliste, damit sind die Einträge ausreichend gekennzeichnet.

TODO 15 : Die Optionen `super` und `long` schreiben je eine Tabelle, entweder als `supertabular` oder als `longtable` -> diese werden derzeit nicht korrekt erkannt

Es muss nichts umdefiniert werden. Ein Umdefinieren des `\glositem` muss nicht stattfinden, da intern auf eine Definitionsliste zu gegriffen wird, was der Autorin von der inhaltlichen Aussage her angemessen erscheint

Glossareinträge und Referenzen

Die im Text verwandten Verweise auf ein Glossareintrag werden mit Hilfe der Funktion `\hyperlink` des `hyperref`-Paketes gesetzt. So dass sie bereits zuverlässig erkannt werden.

Abkürzungen und Akronyme

Eine mögliche Auszeichnung und Anwendungsmöglichkeiten für Akronyme und Abkürzungen stellt das Paket glossary zur Verfügung. Es ermöglicht das Anlegen eines Abkürzungsverzeichnisses und eine Referenzierung der Langform sowie der Kurzform über kurze Befehle.

Dabei sollte für die Kurzform, jeweils die Langform in die PDF-Struktur übernommen werden, so dass assistive Technologien darauf Zugriff haben.

```
1848 \ifthenelse{\boolean{@Access@pdf}}{%
1849   \ifpackageloaded{glossary}{%
1850     \let\originalnewacronym\newacronym%
1851     \renewcommand{\newacronym}[4] []{%
1852       %%% Originaldefinition
1853       \ifthenelse{\equal{#1}{}}{\renewcommand{\@acrnmcmd{#2}}{%
1854         \renewcommand{\@acrnmcmd{#1}}%
1855         \xdef\expansion{#3}%
1856         \@ifundefined{\@acrnmcmd}{%
1857           \expandafter\newcommand\csname\@acrnmcmd short\endcsname{%
1858             \protect\pdfliteral{/Span <</E (\expansion)>> BDC}%
1859           #2%
1860           \protect\pdfliteral{EMC}%
1861           \protect\glxsxspace}%
1862           \expandafter\newcommand\csname\@acrnmcmd @nx@short\endcsname{%
1863             \protect\pdfliteral{/Span <</E (\expansion)>> BDC}%
1864           #2%
1865           \protect\pdfliteral{EMC}}%
1866           \expandafter\newcommand\csname\@acrnmcmd long\endcsname{%
1867             #3\protect\glxsxspace}
1868           \expandafter\newcommand\csname\@acrnmcmd @nx@long\endcsname{#3}
1869           \def\@acrn@entry{#4}%
1870           {%
1871             \expandafter\@gls@getdescr\expandafter{\@acrn@entry}%
1872             \let\glo@desc\glo@desc%
1873             \def\glo@long{#3}%
1874             \@onelevel@sanitize\glo@long
1875             \def\glo@short{\noexpand\acronymfont{#2}}%
1876             \@onelevel@sanitize\glo@short
1877             \expandafter\protected@xdef\expandafter{\@acrn@entry}{\@acrn@entry}%
1878             \expandafter\protected@xdef\expandafter{\@acrn@entry}{\@acrn@entry}%
1879           }%
1880           \@acr@addtolist{\@acrn@entry}%
1881           \@glo@tb=\expandafter{\@acrn@entry}%
1882           \protected@edef\@acr@gl@entry{name={\@acrn@entry},%
1883             format=gl@numformat,sort={\@acrn@entry},\the\@glo@tb,%
1884             description={\@acrn@entry}}%
1885           \@glo@tb=\expandafter{\@acr@gl@entry}%
1886           \newboolean{\@acrn@entry first}\setboolean{\@acrn@entry first}{true}
1887           \expandafter\protected@edef\csname \@acrn@entry\endcsname{%
1888             \noexpand\ifstar{\csname @s@\@acrn@entry\endcsname}{%
1889             \csname \@acrn@entry\endcsname}}
```

```

1890 \ifglshyperacronym % hyperlinks
1891 \expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
1892 \noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
1893 \csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
1894 (\noexpand\xacronym{the\@glo@tb}}{%
1895 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1896 })\noexpand\unsetacronym{\@acrnmcmd}%
1897 }{\noexpand\xacronym{the\@glo@tb}}{%
1898 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1899 \noexpand\@acrnmins}}\noexpand\glxsxspace}
1900 \expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
1901 \noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
1902 \noexpand\expandafter\noexpand\MakeUppercase
1903 \csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
1904 (\noexpand\xacronym{the\@glo@tb}}{%
1905 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1906 })%
1907 \noexpand\unsetacronym{\@acrnmcmd}}{%
1908 \noexpand\xacronym{the\@glo@tb}}{%
1909 \noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
1910 \csname\@acrnmcmd @nx@short\endcsname}%
1911 \noexpand\@acrnmins}}\noexpand\glxsxspace}
1912 \else % no hyperlinks
1913 \expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
1914 \noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
1915 \csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
1916 (\noexpand\xacronym{the\@glo@tb}}{%
1917 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1918 })\noexpand\unsetacronym{\@acrnmcmd}%
1919 }{\noexpand\xacronym{the\@glo@tb}}{%
1920 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1921 \noexpand\@acrnmins}}%
1922 \noexpand\glxsxspace}
1923 \expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
1924 \noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
1925 \noexpand\expandafter
1926 \noexpand\MakeUppercase
1927 \csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
1928 (\noexpand\xacronym{the\@glo@tb}}{%
1929 \noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
1930 })%
1931 \noexpand\unsetacronym{\@acrnmcmd}}{%
1932 \noexpand\xacronym{the\@glo@tb}}{%
1933 \noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
1934 \csname\@acrnmcmd @nx@short\endcsname}%
1935 \noexpand\@acrnmins}}\noexpand\glxsxspace}
1936 \fi
1937 }{%
1938 \PackageError{glossary}{Command '\expandafter\string
1939 \csname\@acrnmcmd\endcsname' already defined}{%
1940 The command name specified by \string\newacronym already exists.}}
1941 %%% Originaldefinition

```

3.13.7 Das booktabs-Paket

Das booktabs-Paket stellt vier neue Befehle für Tabellenlinien zur Verfügung. Bei der Definition wird wiederum auf das Makro `\hrule` zurück gegriffen, so dass eine Auszeichnung als Artefakt bereits erledigt wird.

3.13.8 Das hyperref-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard- \LaTeX -Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.9 Das caption-Paket

Das caption-Paket kann mit seinen möglichen Konfigurationsparameter ohne Einschränkung verwendet werden. Die alte Version das caption2-Paket ist obsolet und sollte nicht mehr verwendet werden.

3.13.10 Das tabularx-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard- \LaTeX -Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.11 Das longtabular-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard- \LaTeX -Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.12 Das color-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard- \LaTeX -Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.13 Das theorem-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard- \LaTeX -Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.14 Das thmbox-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard-L^AT_EX-Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.15 Das listings-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard-L^AT_EX-Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden.

3.13.16 Das scrpage2-Paket

Die Nutzung dieses Pakets ist unter Vorsicht zu genießen. Die Standard-L^AT_EX-Befehle funktionieren auch unter Verwendung des Paketes. Paketeigene Erweiterungen sind größtenteils noch nicht implementiert. Sie konnten bisher nicht vollständig getestet werden. Bei der Nutzung von scrpage2 kommt es zu Problemen bei der Umsetzung des Inhaltsverzeichnisses (TableOfContent).

Literaturverzeichnis

- [Ado04] ADOBE SYSTEMS INC. (Herausgeber): *PDF Reference – Adobe Portable Document Format Version 1.6*. Addison-Wesley, 5. Auflage, 2004. http://www.adobe.com/devnet/pdf/pdf_reference.html.
- [BCJ⁺00] BRAMS, JOHANNES, DAVID CARLISLE, ALAN JEFFREY, LESLIE LAMPORT, FRANK MITTELBACH, CHRIS ROWLEYA und RAINER SCHÖPF: *The L^AT_EX₂ ϵ Sources*. Technischer Bericht, The Latex Project, Juni 2000.
- [Sch07a] SCHALITZ, BABETT: *Accessibility-Erhöhung in LaTeX-Dokumenten*. Diplomarbeit, Technische Universität Dresden, Informatik Fakultät, April 2007.
- [Sch07b] SCHALITZ, BABETT: *Autorenanleitung zur Erstellung von L^AT_EX-Dokumenten hoher Accessibility*. Technische Universität Dresden, Fakultät Informatik, Version 1 Auflage, April 2007.