Das accessibility-Paket

Babett Schalitz

Version 2.0.3, 8. November 2019

Inhaltsverzeichnis

1 Einleitung

Das accessibility-Paket bietet die Möglichkeit „Tagged PDF“ zu erstellen, dass 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 Sehbeeintrüchtigung im Dokument besser navigieren kann, indem er eine interessante überschrift direkt anspringt.
- Prinzipiell konnen 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, dass 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

 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, dass 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 verhölt es sich mit den Optionen `untagged` und `tagged`. Gibt man keine Optionen an, so wird

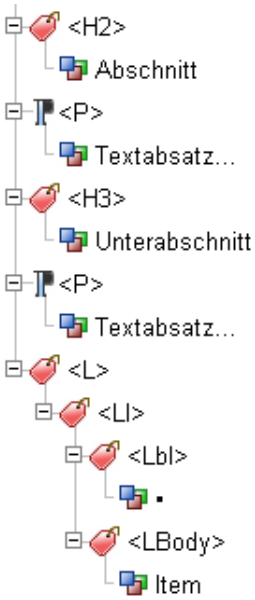
ein PDF mit den Standardoptionen erzeugt. D. h. es wird Tagged PDF mit einer geschachtelten Struktur erzeugt.

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

Die Latex-Struktur

```
\section{Abschnitt}
  Textabsatz...
\subsection{Unterabschnitt}
  Textabsatz...
\begin{itemize}
  \item Item
\end{itemize}
```

Struktur mit `flatstructure`



Struktur mit `highstructure`

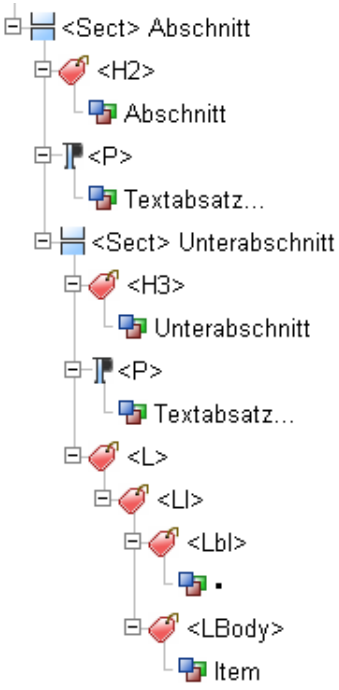


Abbildung 2.1: Erläuterungen zu flachen und strukturierten Variante

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 Wurzelelement (`/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 geschrieben Objekte nicht. Es kommt zu doppelten ID auf einer Seite, was die eindeutige Zuordnung stört und zahlreiche Fehler birgt. Folgefehler dieses Problems künnen durch die durchgehenden Nummerierung beseitigt werden.

```
\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.

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

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

```
\gdef\DocumentLanguage{}%
\gdef\ActualLanguage{}%
\newif\ifLanguageDiff \global\LanguageDifffalse%
\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.

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

3.1.3 Definition der Optionen

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

```
\DeclareOption{flatstructure}{\global\PDFDetailedStructurefalse}%
\DeclareOption{highstructure}{\global\PDFDetailedStructuretrue}%
\DeclareOption{tagged}{\setboolean{@tagged@pdf}{true}}%
\DeclareOption{untagged}{\setboolean{@tagged@pdf}{false}}%
\DeclareOption*{%
   \PackageWarning{accessibility}{Unknown Option \CurrentOption}}%
\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.

```
\ifthenelse{\isundefined{\pdfoutput}}{%
  %latex with dvips
  \setboolean{@right@pdfversion}{false}%
  }{\ifthenelse{\number\pdfoutput<1}{%
      %pdflatex in DVI mode
      \setboolean{@right@pdfversion}{false}%
      }{%pdflatex in PDF mode
      \ifthenelse{\pdftexversion<120}{%
          \PackageError{accessibility}%
          {pdfTeX/pdfLaTeX version >= 1.20 required for direct PDF outut}%
          {Try to install a more recent version!}%
      }{%
      %It is the right version
      \setboolean{@right@pdfversion}{true}%
    }%
  }%
}
```

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

```
\ifthenelse{\boolean{@right@pdfversion} \and \boolean{@tagged@pdf}}{%
      \setboolean{@Access@pdf}{true}%
}{%
      \setboolean{@Access@pdf}{false}%
}
```

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.

```
\newboolean{@KOMAScriptClass} \setboolean{@KOMAScriptClass}{false}%

\@ifclassloaded{scrreprt} {\setboolean{@KOMAScriptClass}{true}}{}%
\@ifclassloaded{scrbook}  {\setboolean{@KOMAScriptClass}{true}}{}%
\@ifclassloaded{scrartcl} {\setboolean{@KOMAScriptClass}{true}}{}%
\ifthenelse{\boolean{@KOMAScriptClass}}{%
                \PackageInfo{accessibility}{KOMAscript Klasse}}{}%

\newboolean{@StandardClass} \setboolean{@StandardClass}{false}%

\@ifclassloaded{report} {\setboolean{@StandardClass}{true}}{}%
\@ifclassloaded{book}   {\setboolean{@StandardClass}{true}}{}%
\@ifclassloaded{article}{\setboolean{@StandardClass}{true}}{}%

\ifthenelse{\boolean{@StandardClass}}{%
```

```
65                \PackageInfo{accessibility}{Standardklasse}}{}%
```

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 Objektnummer 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 darrunterliegendes 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 abzulegen.

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

```
\newcount\@stackdepth \@stackdepth=0%
\def\accessPushStack#1#2#3#4{%
   \ifnum \@stackdepth >15\relax%
        \PackageWarning{accessibility}{too deep}%
   \else%
        \global\advance\@stackdepth\@ne%
   \fi%
    {\expandafter\xdef\csname StackA\romannumeral\the\@stackdepth\endcsname{#1}}%
   {\expandafter\xdef\csname StackB\romannumeral\the\@stackdepth\endcsname{#2}}%
   {\expandafter\xdef\csname StackC\romannumeral\the\@stackdepth\endcsname{#3}}%
   {\expandafter\xdef\csname StackD\romannumeral\the\@stackdepth\endcsname{#4}}%
}%
```

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

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

```
\def\accessPopStack#1#2#3#4{%
  \ifnum \the\@stackdepth <1\relax%
     \global\let#1\empty%
     \global\let#2\empty%
     \global\let#3\empty%
     \global\let#4\empty%
  \else%
     \xdef#1{\csname StackA\romannumeral\the\@stackdepth\endcsname}%
     \xdef#2{\csname StackB\romannumeral\the\@stackdepth\endcsname}%
     \xdef#3{\csname StackC\romannumeral\the\@stackdepth\endcsname}%
     \xdef#4{\csname StackD\romannumeral\the\@stackdepth\endcsname}%
   %Variablen wieder leeren
   {\expandafter\xdef\csname StackA\romannumeral\the\@stackdepth\endcsname{}}%
   {\expandafter\xdef\csname StackB\romannumeral\the\@stackdepth\endcsname{}}%
   {\expandafter\xdef\csname StackC\romannumeral\the\@stackdepth\endcsname{}}%
   {\expandafter\xdef\csname StackD\romannumeral\the\@stackdepth\endcsname{}}%
     \global\advance\@stackdepth\m@ne%
  \fi%
}%
```

`\accessReadTopStack` Zum anderen wird wären der Erzeugung von Blattknoten ein lesender Zugriff auf das oberste Stackelement benötigt. Somit kann die Objektreferenz in dass Kinderfeld des Elternelementes eingetragen werden und eine Referenzierung des Elternobjektes wird möglich.

Parameter #1 Type #2 ObjNum #3 KidsField

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

```
      \global\let#3\empty%
   \else%
      \xdef#1{\csname StackA\romannumeral\the\@stackdepth\endcsname}%
      \xdef#2{\csname StackB\romannumeral\the\@stackdepth\endcsname}%
      \xdef#3{\csname StackC\romannumeral\the\@stackdepth\endcsname}%
   \fi%
}%
```

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

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

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.

```
{%andere Ebenen Schließen
   \accessPopStack\StructElem\Objnum\KidsArray\Title %runterholen
   \ifx \StructElem\empty%
     \PackageWarning{accessibility}{empty \string\PopStack ?}%
   \else%
     %\convertLanguageInCode{\languagename}%
     \gdef\TitleHelp{}%
     \if \Title\empty \else%
         \gdef\TitleHelp{/T (\Title)}%
     \fi%
      \ifnum \@stackdepth <1\relax%
           \xdef\ParentElem{\theStructTree}%
           \xdef\ParentArray{Karray}%
      \else%
           \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.

```
\newcommand{\closeUntilPDFStruct}[1]{%
      \ifTextActive%
          \endPDFMarkContent%
          \endPDFTextObj%
      \fi%
     %Schleife
     \@bool=0%
        \sectionInDepth{#1}{@elemi}%
        \ifnum \@elemi=100 \@bool=1 \fi%
        \global\advance\@elemi\m@ne%
     \@whilenum \@bool =0\do{%
        \accessReadTopStack\StructElem\Objnum\KidsArray%
        \sectionInDepth{\StructElem}{@elem}%
        \ifthenelse{\@elem >\@elemi}{%
             \endPDFStructObj%
        }{%
             \@bool=1%
        }%
        \if #1\empty \@bool=1 \fi%
        \ifthenelse{\equal{#1}{\StructElem}}{\@bool=1}{}%
     }%
}
```

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.

```
\xdef\TextType{}%
\newcounter{TextObjNum}%
\xdef\TextArray{}%
\newif\ifTextActive \TextActivefalse%
\newif\ifSpezialTextActive \SpezialTextActivefalse%
```

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

```
\newenvironment*{PDFTextObj}{%
  %altes Textobj beenden, immer max. ein Textobj aktiv
  \ifTextActive \endPDFMarkContent\endPDFTextObj\fi%
  %neues anfangen
  \global\TextActivetrue%
  \pdfobj reserveobjnum% Objektnummer reservieren
  \setcounter{TextObjNum}{\pdflastobj}%
  \xdef\TextArray{}%
  \xdef\TextType{P}% kein TextTxp --> P
}%
```

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.

```
{%
 \ifTextActive%
  \ifnum \@stackdepth <1\relax%
       \xdef\ParentElem{\theStructTree}%
       \xdef\ParentArray{Karray}%
       %\PackageWarning{accessibility}{stackdepth<1}%
  \else%
       \accessReadTopStack\ParentStructElem\ParentElem\ParentArray%
  \fi%
  %\convertLanguageInCode{\languagename}%
  \immediate \pdfobj useobjnum\theTextObjNum{<</Type /StructElem %
     /P \ParentElem \space 0 R %
     /C /Normal %
     /K [\TextArray] %
     /S /\TextType %
    \space\LanguageCode>>}%
  \pdfrefobj \theTextObjNum%
  \expandafter\xdef\csname \ParentArray\endcsname{%
       \csname \ParentArray\endcsname \space \theTextObjNum\space 0 R}%
  \global\TextActivefalse%
 \fi%
}
```

`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.

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

Der Beginn eines speziellen Textobjektes ist analog dem eines Normalen.

Parameter #1 StructType

```
\newenvironment*{PDFSpezialTextObj}[1]{%
  \ifTextActive \endPDFMarkContent\endPDFTextObj\fi%
  %neues anfangen
  \global\SpezialTextActivetrue%
  \pdfobj reserveobjnum% Objektnummer reservieren
  \setcounter{TextObjNum}{\pdflastobj}%
  \xdef\TextArray{}%
  \xdef\TextType{#1}%
}%
```

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

```
{%
  %\gdef\LanguageHelp{}%
  %\ifLanguageDiff%
  %    \gdef\LanguageHelp{\LanguageCode}%
  %\fi%
  %  \convertLanguageInCode{\languagename}%
    \gdef\AltHelp{}%
  \ifthenelse{\equal{\altAttr}{}}{}{%
     \gdef\AltHelp{/Alt(\altAttr)}%
  }%
  \gdef\TitleHelp{}%
  \ifthenelse{\equal{\titleAttr}{}}{}{%
     \gdef\TitleHelp{/T(\titleAttr)}%
  }%
  \ifnum \@stackdepth <1\relax%
       \xdef\ParentElem{\theStructTree}%
       \xdef\ParentArray{Karray}%
  \else%
       \accessReadTopStack\ParentStructElem\ParentElem\ParentArray%
  \fi%
  \immediate \pdfobj useobjnum\theTextObjNum{<</Type /StructElem %
     /P \ParentElem \space 0 R %
     /C /Normal %
     /K [\TextArray] %
     /S /\TextType %
    \space\LanguageCode %
    \space\TitleHelp %
    \space\AltHelp>>}%
  \pdfrefobj \theTextObjNum%
  \expandafter\xdef\csname \ParentArray\endcsname{%
      \csname \ParentArray\endcsname \space \theTextObjNum\space 0 R}%
  \global\SpezialTextActivefalse%
  \xdef\TextType{}%
  \EveryparReset%
  \xdef\altAttr{}% wieder leeren
  \xdef\titleAttr{}% wieder leeren
}
```

3.2.4 Elemente auf Zeilenebene

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

```
\xdef\lastEveryparType{}%
\xdef\HelpBool{}%
\xdef\Type{}%
\newcounter{PDFReferenceObjNum}%
\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.

```
\newenvironment*{PDFInlineObjInText}[1]{%
  %alte einstellung merken
  \xdef\lastEveryparType{\everyparStructElem}%
  \xdef\HelpBool{\InlineObj}%
  \xdef\Type{#1}%
  \ifthenelse{\equal{\Type}{Reference} \or \equal{\Type}{Link}}{%
    \pdfobj reserveobjnum%
    \setcounter{PDFReferenceObjNum}{\pdflastobj}%
     \setcounter{ObjNum}{\theTaggedObj}%
     \EveryparConfig{#1}{obj}%
  }{%
     \EveryparConfig{#1}{true}%
  }%
  \PDFMarkContent% kein everypar
}%
```

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.

```
{%
  \endPDFMarkContent%
  \ifthenelse{\equal{\Type}{Reference} \or \equal{\Type}{Link}}{%
    \xdef\ReferenceArray{<</Type /MCR /Pg \pdfpageref\thepage %
         \space \space 0 R /MCID \theObjNum>>}%
    \ifthenelse{\pdftexversion>139}{%
      \setcounter{ObjHelp}{\pdflastlink}%%<</Type /OBJR /Obj 600 0 R>>
      %\PackageWarning{accessibility}%
      %{Objektnummer vom letzten Link: \theObjHelp}%
      \ifthenelse{\theObjHelp>0}{%
        \xdef\ReferenceArray{\ReferenceArray\space %
              <</Type /OBJR /Obj \theObjHelp\space 0 R>>}%
      }{}%
```

```
    }{}%
    \writeComplexTextObj{\thePDFReferenceObjNum}%
          {\ReferenceArray}%
          {/Reference}{\theTextObjNum}{NoPage}%
    \xdef\TextArray{\TextArray \theObjHelp\space 0 R \space}%
  }{}%
  % alte einstellung wiederherstellen
  \EveryparConfig{\lastEveryparType}{\HelpBool}%
  \PDFMarkContent%
}
```

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 `\EyeryparReset` dienen der Flexibilisierung von MarkContent, 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

```
\newcommand{\EveryparConfig}[2]{%
    \xdef\everyparStructElem{#1}%
    \xdef\InlineObj{#2}%
}
```

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

```
\newcommand{\EveryparReset}{%
    \xdef\everyparStructElem{P}%
    \xdef\InlineObj{false}%
}
```

`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 allgemeinem 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.

```
\newif\ifMarkContentOpen \MarkContentOpenfalse%
\xdef\everyparStructElem{P}%
\xdef\InlineObj{false}%
```

Zunächst wird eine eventuell geöffnete Markierung abgeschossen. 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.

```
\newenvironment{PDFMarkContent}{%
  \ifMarkContentOpen \endPDFMarkContent\fi%
  \global\MarkContentOpentrue%
  \ifthenelse{\equal{\everyparStructElem}{}}{\EveryparReset}{}%
  \pdfliteral{/\everyparStructElem\space <</MCID \theTaggedObj>> BDC}%
  \ifthenelse{\equal{\InlineObj}{false}}{% Text
    %\ifthenelse{\equal{\themypage}{\thepage}}{}{%
       %\PackageWarning{accessibility}{unterschiedlich %
       %\thepage<>\themypage; MCID \theTaggedObj}
    %}%
    \xdef\TextArray{\TextArray \space <</Type /MCR %
        %/Pg \pdfpageref\thepage \space \space 0 R %
        /Pg \pdfpageref\themypage \space \space 0 R %
        /MCID \theTaggedObj>>}%
  }{}%
  \ifthenelse{\equal{\InlineObj}{true}}{%  inline-Objekt
    \writePDFLeafObj{\theTaggedObj}{\everyparStructElem}%
  }{}%
  \stepcounter{TaggedObj}%
}%
```

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.

```
{%
 \ifMarkContentOpen%
    \pdfliteral{EMC}%
    \global\MarkContentOpenfalse%
 \fi%
}
```

Zur besseren übersichtlichkeit und wurde das Makro `\writePDfLeafObj` ausgelagert. Es kann so des Weiteren auch an anderer Stele 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

```
\newcommand*{\writePDFLeafObj}[2]{%
  %\gdef\LanguageHelp{}%
  %\ifLanguageDiff%
  %    \gdef\LanguageHelp{\LanguageCode}%
  %\fi%
  %  \convertLanguageInCode{\languagename}%
  \gdef\AltHelp{}%
  \ifthenelse{\equal{\altAttr}{}}{}{%
     \gdef\AltHelp{/Alt(\altAttr)}%
  }%
  \gdef\TitleHelp{}%
  \ifthenelse{\equal{\titleAttr}{}}{}{%
     \gdef\TitleHelp{/T(\titleAttr)}%
  }%
  \immediate \pdfobj {<</Type /StructElem %
    /P \theTextObjNum \space 0 R %
    /C /Normal %
    /K <</Type /MCR %
         /Pg \pdfpageref\thepage \space \space 0 R %
         /MCID #1>> %
    /S /#2 %
    \space\LanguageCode %
    \space\TitleHelp %
    \space\AltHelp>>}%
  \setcounter{ObjHelp}{\pdflastobj}%
  \pdfrefobj \pdflastobj%
  \xdef\TextArray{\TextArray \space \theObjHelp\space 0 R}%
  \xdef\altAttr{}%
  \xdef\titleAttr{}%
}
```

`\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 Objektnummer 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 Objektnummer 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 StructurType #4 ParentObj #5 (no)page

```
\newcommand*{\writeComplexTextObj}[5]{%
  \ifthenelse{\equal{#5}{Page}}{% Seitenreferenz angeben
    \immediate \pdfobj useobjnum#1{<</Type /StructElem %
      /P #4\space 0 R %
      /Pg \pdfpageref\thepage \space \space 0 R %
      /C /Normal %
      /K [#2] %
      /S #3 \space\LanguageCode>>}%
    }{%keine Seitenreferenz angeben
    \immediate \pdfobj useobjnum#1{<</Type /StructElem %
      /P #4\space 0 R %
      /C /Normal %
      /K [#2] %
      /S #3>>}%
  }%
  \setcounter{ObjHelp}{\pdflastobj}%
  \pdfrefobj \pdflastobj%
}
```

3.3 Erkennen von Absätzen

Da Textabsätze in LaTeX 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 ausgesetz 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.

```
  %\newcounter{ParCounter}[page]%
\newcounter{ParCounter}%
\newif\ifnumberingpars \numberingparstrue%
\newif\ifnpar \npartrue%
```

Nun wird die Funktion `\everypar` tatsächlich umdefiniert. 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.

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

```
  \newtoks\npeverypar%
  \npeverypar{}%
  % Call everypar with the argument extended with the toks
  \def\everypar#1{%
    \originaleverypar{#1\ifnumberingpars\the\npeverypar\fi}}%
  \npeverypar{%
    \ifnpar{%
      \stepcounter{ParCounter}%
     %\pdfliteral{/Artifact BMC}%
     % \llap{\small\arabic{ParCounter}\qquad}%
     %\pdfliteral{EMC}%
      \ifSpezialTextActive\else \PDFTextObj \fi%
      \PDFMarkContent%
    }\fi%
  }%
}{}
```

3.4 Dokumentbeginn

Strukturbaum eröffnen.

```
\AtBeginDocument{%
\PDFStructObj{Document}{\empty}%
\everypar{}%
}
```

3.5 Dokumentende

Strukturbaum abschließen

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

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

```
  \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.

```
  \newcounter{ClassMap}%
  \pdfobj reserveobjnum% neues Objekt Reservieren
  \setcounter{ClassMap}{\pdflastobj}%
  \immediate \pdfobj useobjnum \theClassMap{<<%
      /Normal <</O /Layout /EndIndent 0.0 %
                /SpaceAfter 0.0 /SpaceBefore 0.0 %
                /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 % Bibligraphy
462            /BibItem /LI % BibligraphyItem
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.

```
\pdfcatalog{%
    /StructTreeRoot \theStructTree\space 0 R%
    /MarkInfo <</Marked true /LetterspaceFlags 0>>%
    %/PieceInfo<</MarkedPDF>>%
    %/MarkInfo <</Marked true>>%
    %/Metadata \theStructTree\space 0 R%
    }%
```

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

```
\ifthenelse{\boolean{ACCESSProblems}}{%
   \PackageWarningNoLine{accessibility}{%
      @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\MessageBreak
      @@ There were non-defined Figure Alt-Tags! @@\MessageBreak
      @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@}%
}{}%
}%
}{}
```

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 gewährleistest.

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 gesieht 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.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\linenopenalty\linenopenalty=-100000%
\mathchardef\linenopenaltypar=32000%
%
%%%%%%%%%%%%%%%Verändern der Outputroutine%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\@tempa\output%
  \newtoks\output%
  \let\@AC@output\output%
  \output=\expandafter{\the\@tempa}%
  %
  \@tempa={%
    % LineNoTest
    \let\@@par\@@@par%
    \ifnum\interlinepenalty<-\linenopenaltypar%
       \advance\interlinepenalty-\linenopenalty%
       \@AC@nobreaktrue%
    \fi%
    \@tempswatrue%
    \ifnum\outputpenalty>-\linenopenaltypar\else%
       \ifnum\outputpenalty>-188000\relax%
         \@tempswafalse%
         \fi%
       \fi%
    \if@tempswa%
         % LineNoLaTeXOutput
         \ifnum \holdinginserts=\thr@@ %
           \global\holdinginserts-\thr@@ %
           \unvbox\@cclv %
           \ifnum \outputpenalty=\@M \else \penalty\outputpenalty \fi %
         \else%
           \if@twocolumn \let\@makecol\@AC@makecol \fi%
           \the\@AC@output %
           \ifnum \holdinginserts=-\thr@@ %
             \global\holdinginserts\thr@@ \fi %
         \fi%
    \else  %
       %MakeLineNo
       \boxmaxdepth\maxdimen\setbox\z@\vbox{\unvbox\@cclv}%
       \@tempdima\dp\z@ \unvbox\z@%
       \sbox\@tempboxa{\hb@xt@\z@{\makeLineNumber}}%
       \stepcounter{linenumber}%
       \stepcounter{abslinenumber}%
       \ht\@tempboxa\z@ \@AC@depthbox %
       \count@\lastpenalty %
       \ifnum\outputpenalty=-\linenopenaltypar %
         \ifnum\count@=\z@ \else %
```

```
            \xdef\@AC@parpgbrk{%
              \penalty\the\count@%
              \global\let\noexpand\@AC@parpgbrk%
                        \noexpand\@AC@screenoff@pen}%
          \fi%
        \else%
          \@tempcnta\outputpenalty%
          \advance\@tempcnta -\linenopenalty%
          \penalty \ifnum\count@<\@tempcnta \@tempcnta \else \count@ \fi %
        \fi%
      \fi%
    }%
    \def\@AC@nobreaktrue{\let\if@nobreak\iftrue} %
    %
    \def\@AC@depthbox{%
      \dp\@tempboxa=\@tempdima%
      \nointerlineskip \kern-\@tempdima \box\@tempboxa} %
    %
    \def\@AC@screenoff@pen{%
      \ifdim\lastskip=\z@ %
        \@tempdima\prevdepth \setbox\@tempboxa\null %
        \@AC@depthbox                           \fi}%
    %
    \global\let\@AC@parpgbrk\@AC@screenoff@pen %
}{}%
    %
    %%%%%%%%%%%%%Initializieren der Variablen%%%%%%%%%%%%%%%%%%%%%%%%
    %
    %Zeilennummer
    \newcounter{linenumber}%
    \newcounter{abslinenumber}%
    %Seitennummer
    \newcount\c@AC@truepage %
    \global\advance\c@AC@truepage\@ne %mit eins beginnen
    %\g@addto@macro\cl@page{\global\c@AC@truepage\c@page}%
    \g@addto@macro\cl@page{\global\advance\c@AC@truepage\@ne}%
    \@addtoreset{AC@truepage}{@ckpt}%
    %
    \newcounter{mypage}%
    \setcounter{mypage}{\@ne}%
    \g@addto@macro\cl@page{\global\c@mypage\c@page}%
    \@addtoreset{mypage}{@ckpt}%
    %
    %%%%%%%%%%%%%Schreiben der Zeilennummmern%%%%%%%%%%%%%%%%%%%%%%%%
\ifthenelse{\boolean{@Access@pdf}}{%
    \def\makeLineNumber{%
      \protected@write\@auxout{}{\string\@AC{\the\c@linenumber}%
                                {\noexpand\the\c@AC@truepage}}%
      \testNumberedPage%
      %Schreibt die Zeilennummern
      %\hss{\normalfont\tiny\sffamily\thelinenumber\quad}%
    }%
```

```
}{}%
  %
  %%%%%%%%%%%%%Absatz
\newif\ifLastLineStop \LastLineStopfalse%
\newcounter{LastPar}%

\newcommand{\EndPage}[1]{%
    \ifMarkContentOpen%
       \endPDFMarkContent%
       \global\LastLineStoptrue%
       \setcounter{LastPar}{\noexpand\theParCounter}%
    \else%
       \global\LastLineStopfalse%
    \fi%
    \if@twocolumn \else%
       \stepcounter{mypage}%
    \fi%
}%

\newcommand{\BeginPage}{%
   \ifLastLineStop%
     \ifnum \theParCounter=\theLastPar%
        \pdfliteral{/P <</MCID \theTaggedObj>> BDC}%
        \global\MarkContentOpentrue%
         \xdef\TextArray{\TextArray \space <</Type /MCR %
                /Pg \pdfpageref\the\c@AC@truepage \space \space 0 R %
                /MCID \theTaggedObj>>}%
        \stepcounter{TaggedObj}%
     \fi%
   \fi%
}%

\ifthenelse{\boolean{@Access@pdf}}{%
  \let\original@startcolumn\@startcolumn%
  \renewcommand{\@startcolumn}{%
    \BeginPage%
    \original@startcolumn%
  }%
}{}%

  %%%%%%%%%%%%%Berichtigung der Seitennummer%%%%%%%%%%%%%%%%%%%%%%%%
\ifthenelse{\boolean{@Access@pdf}}{%
  \def\LastNumberedPage{first}%
  \def\AC@Pfirst{\nextAC\relax}%
  %
  \let\lastAC\relax  % compare to last line on this page
  \let\firstAC\relax % compare to first line on this page
  \let\pageAC\relax  % get the page number, compute the linenumber
  \let\nextAC\relax  % move to the next page
  %
  \AtEndDocument{\let\@AC\@gobbletwo} %
  %
```

```
\def\@AC#1#2{{\expandafter\@@AC%
                  \csname AC@P#2C\@AC@column\expandafter\endcsname%
                  \csname AC@PO#2\endcsname%
                  {#1}{#2}}}%
%
\def\@@AC#1#2#3#4{\ifx#1\relax%
  \ifx#2\relax\gdef#2{#3}\fi%
    \expandafter\@@@AC\csname AC@P\LastNumberedPage\endcsname#1%
    \xdef#1{\lastAC{#3}\firstAC{#3}%
            \pageAC{#4}{\@AC@column}{#2}\nextAC\relax}%
  \else%
    \def\lastAC##1{\noexpand\lastAC{#3}}%
    \xdef#1{#1}%
  \fi%
  \xdef\LastNumberedPage{#4C\@AC@column}%
}%
%
\def\@@@AC#1#2{{\def\nextAC##1{\noexpand\nextAC\noexpand#2}%
                \xdef#1{#1}}}%
%
\def\NumberedPageCache{\AC@Pfirst}%
%
\def\testLastNumberedPage#1{%
  \ifnum#1<\c@linenumber%
    \let\firstAC\@gobble%
  \fi%
  \ifnum#1=\c@linenumber%
    \EndPage{#1}%
  \fi%
}%
%
\def\testFirstNumberedPage#1{%
  \ifnum#1>\c@linenumber%
     \def\nextAC##1{\testNextNumberedPage\AC@Pfirst}%
  \else%
    \let\nextAC\@gobble%
    \def\pageAC{\gotNumberedPage{#1}}%
  \fi%
}%
%
\long\def \@gobblethree #1#2#3{}%
%
\def\testNumberedPage{%
  \let\lastAC\testLastNumberedPage%
  \let\firstAC\testFirstNumberedPage%
  \let\pageAC\@gobblethree%
  \let\nextAC\testNextNumberedPage%
  \NumberedPageCache%
}%
%
\def\testNextNumberedPage#1{%
 \ifx#1\relax%
```

```
      \global\def\NumberedPageCache{\gotNumberedPage0000}%
     \PackageWarning{accessibility}{Changed paragraphs, rerun to get it right}%
    \else%
      \global\let\NumberedPageCache#1%
    \fi%
    \testNumberedPage%
   }%
   %
   \def\gotNumberedPage#1#2#3#4{%
     \ifodd \if@twocolumn #3\else #2\fi\relax\fi%
     \advance\c@linenumber\@ne % Nummerierung ab 1 sonst ab 0
     \advance\c@linenumber-#4\relax%
   }%
 }{}%
   %
   %%%%%%%%%%%%%%%Spaltenerkennung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 \ifthenelse{\boolean{@Access@pdf}}{%
   \def\@AC@col{\def\@AC@column} %
   \@AC@col{1}%
   %
   \AtBeginDocument{\let\@AC@orig@makecol\@makecol} %
   %
   \def\@AC@makecol{%
      \@AC@orig@makecol
      \setbox\@outputbox \vbox{%
         \boxmaxdepth \@maxdepth%
         \protected@write\@auxout{}{%
             \string\@AC@col{\if@firstcolumn1\else2\fi}%
         }%
         \box\@outputbox %
   }}%
 }{}%
   %
   %%%%%%%%%%%%%%%Absatzerkennung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 \ifthenelse{\boolean{@Access@pdf}}{%
   \let\@@@par\@@par%
   \newcount\linenoprevgraf%
   %
   \def\linenumberpar{%
     \ifvmode \@@@par \else %
       \ifinner \@@@par \else%
         \xdef\@AC@outer@holdins{\the\holdinginserts}%
         \advance \interlinepenalty \linenopenalty%
         \linenoprevgraf \prevgraf%
         \global \holdinginserts \thr@@ %
         \@@@par%
         \ifnum\prevgraf>\linenoprevgraf%
           \penalty-\linenopenaltypar%
         \fi%
         \@AC@parpgbrk %
         \global\holdinginserts\@AC@outer@holdins%
         \advance\interlinepenalty -\linenopenalty%
```

```
      \fi%
    \fi}%
  %
  \AtEndOfPackage{%
    \xdef\@AC@outer@holdins{\the\holdinginserts}%
    \let\@@par\linenumberpar%
    \ifx\@par\@@@par\let\@par\linenumberpar\fi%
    \ifx\par\@@@par\let\par\linenumberpar\fi%
  }%
}{}%
  %
  %%%%%%%%%%%%%%Formelbehandlung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\linenomath{%
  \ifnum\interlinepenalty>-\linenopenaltypar
    \global\holdinginserts\thr@@
    \advance\interlinepenalty \linenopenalty
   \ifhmode                                  %
    \advance\predisplaypenalty \linenopenalty
   \fi
    \advance\postdisplaypenalty \linenopenalty
    \advance\interdisplaylinepenalty \linenopenalty
  \fi
  \ignorespaces
}%

\def\endlinenomath{%
    \global\holdinginserts\@AC@outer@holdins %
    \global\@ignoretrue
}

\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifundefined{mathindent}{%
    \let\AC@displaymath\[%
    \let\AC@enddisplaymath\]%
    \renewcommand\[{\begin{linenomath}\AC@displaymath}%
    \renewcommand\]{\AC@enddisplaymath\end{linenomath}}%
    %
    \let\AC@equation\equation%
    \let\AC@endequation\endequation%
    \renewenvironment{equation}%
       {\linenomath\AC@equation}%
       {\AC@endequation\endlinenomath}%
  }{}%
  %
  \let\AC@eqnarray\eqnarray%
  \let\AC@endeqnarray\endeqnarray%
  \renewenvironment{eqnarray}%
     {\linenomath\AC@eqnarray}%
     {\AC@endeqnarray\endlinenomath}%
  %
  \advance\maxdeadcycles 100%
}{}%
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalnewpage\newpage%
  \renewcommand{\newpage}{%
    \endPDFMarkContent%
    \originalnewpage%
  }%
}{}%
```

3.7 Überschriften

Die Latex-Struktur

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

Die PDF-Struktur

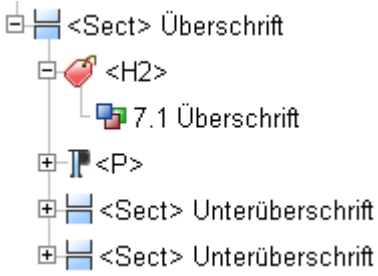


Abbildung 3.1: Struktur für Überschriften

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.

```
\newenvironment{PDFSect}[2]{%
  \ifPDFDetailedStructure%
      \closeUntilPDFStruct{#1}%
      \PDFStructObj{#1}{#2}%
  \fi%
```

```
  \PDFSpezialTextObj{H}%
  \EveryparConfig{H}{false}%
}%
```

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.

```
{%
  \endPDFMarkContent%
  \endPDFSpezialTextObj%
}
```

Nachdem nun die abstrakten Hilfsmakros angelegt sind, können die betroffenen Gliederungsbefehle 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 ist eine Fallunterscheidung nötig.

Umdefinieren des chapter-Befehls

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifundefined{chapter}{% es gibt keine Chapter z.B. in Article-Klassen
    }{%
    \let\originalchapter\chapter%
    \renewcommand{\chapter}{%Sortiert die verschiedenen Aufrufe
      \@ifstar{\originalchapterWithStar}%\chapter*{Beispielkapitel}
      {\@ifnextchar[%]
          {\originalchapterWithTwoOption}%\chapter[BspKap]{Beispielkapitel}
          {\originalchapterWithOption}%\chapter{Beispielkapitel}
      }%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalchapterWithStar}[1]{%
  \PDFSect{Chapter}{#1}\originalchapter*{#1}\endPDFSect}%
\newcommand{\originalchapterWithTwoOption}[2]{%
  \PDFSect{Chapter}{#1}\originalchapter[#1]{#2}\endPDFSect}%
\newcommand{\originalchapterWithOption}[1]{%
  \PDFSect{Chapter}{#1}\originalchapter{#1}\endPDFSect}%
```

addchap Das Umdefinieren des \addchap-Befehls.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifundefined{addchap}{% es gibt keine Chapter z.B. in Article-Klassen
  }{%
    \let\originaladdchap\addchap%
    \renewcommand{\addchap}{%
      \@ifstar{\originaladdchapWithStar}%
      {\@ifnextchar[%]
          {\originaladdchapWithTwoOption}%
          {\originaladdchapWithOption}%
      }%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originaladdchapWithStar}[1]{%
  \PDFSect{Chapter}{#1} \originaladdchap*{#1} \endPDFSect}%
\newcommand{\originaladdchapWithTwoOption}[2]{%
  \PDFSect{Chapter}{#1} \originaladdchap[#1]{#2} \endPDFSect}%
\newcommand{\originaladdchapWithOption}[1]{%
  \PDFSect{Chapter}{#1} \originaladdchap{#1} \endPDFSect}%
```

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifundefined{set@preamble}{% es gibt kein set@preamble%
    }{%  %außerhalb des KOMA-Scripts
    \let\originaluse@preamble\use@preamble%
    \renewcommand{\use@preamble}[1]{%
        \EveryparConfig{P}{true}%
        \originaluse@preamble{#1}%
        \EveryparConfig{H}{false}%
    }%
  }%
}{}
```

3.7.3 Überschriften mit Afterskip

Diese Gliederungsebenen gibt es in allen Dokumentenklassen.

section Umdefinieren des \section-Befehls

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

```
    {\@ifnextchar[%]
        {\originalsectionWithTwoOption}%
        {\originalsectionWithOption}%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalsectionWithStar}[1]%
  {\PDFSect{Section}{#1} \originalsection*{#1} \endPDFSect}%
\newcommand{\originalsectionWithTwoOption}[2]%
  {\PDFSect{Section}{#1} \originalsection[#1]{#2} \endPDFSect}%
\newcommand{\originalsectionWithOption}[1]%
  {\PDFSect{Section}{#1} \originalsection{#1} \endPDFSect}%
```

`subsection` Umdefinieren des `\subsection`-Befehls

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalsubsection\subsection%
  \renewcommand{\subsection}{%
    \@ifstar{\originalsubsectionWithStar}%
    {\@ifnextchar[%]
        {\originalsubsectionWithTwoOption}%
        {\originalsubsectionWithOption}%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalsubsectionWithStar}[1]%
  {\PDFSect{Subsection}{#1} \originalsubsection*{#1} \endPDFSect}%
\newcommand{\originalsubsectionWithTwoOption}[2]%
  {\PDFSect{Subsection}{#1} \originalsubsection[#1]{#2} \endPDFSect}%
\newcommand{\originalsubsectionWithOption}[1]%
  {\PDFSect{Subsection}{#1} \originalsubsection{#1} \endPDFSect}%
```

`subsection` Umdefinieren des `\subsubsection`-Befehls

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalsubsubsection\subsubsection%
  \renewcommand{\subsubsection}{%
    \@ifstar{\originalsubsubsectionWithStar}%
      {\@ifnextchar[%]
        {\originalsubsubsectionWithTwoOption}%
        {\originalsubsubsectionWithOption}%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalsubsubsectionWithStar}[1]%
  {\PDFSect{Subsubsection}{#1} \originalsubsubsection*{#1} \endPDFSect}%
\newcommand{\originalsubsubsectionWithTwoOption}[2]%
  {\PDFSect{Subsubsection}{#1} \originalsubsubsection[#1]{#2} \endPDFSect}%
\newcommand{\originalsubsubsectionWithOption}[1]%
  {\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.

```
\newenvironment{PDFParSect}[2]{%
  %\ifPDFDetailedStructure%
  %    \closeUntilPDFStruct{#1}%
  %    \PDFStructObj{#1}{#2}%
  %\fi%
  \PDFTextObj%
  \EveryparConfig{P}{false}%
  \PDFMarkContent%
}%
```

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.

```
{%
  %\endPDFMarkContent% erst durch everypar
  %\endPDFTextObj%
}
```

`paragraph` Umdefinieren des \paragraph-Befehls

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalparagraph\paragraph%
  \renewcommand{\paragraph}{%
    \@ifstar{\originalparagraphWithStar}%
    {\@ifnextchar[%]
        {\originalparagraphWithTwoOption}%
        {\originalparagraphWithOption}%
    }%
```

```
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalparagraphWithStar}[1]%
  {\PDFParSect{Paragraph}{#1} \originalparagraph*{#1} \endPDFParSect}%
\newcommand{\originalparagraphWithTwoOption}[2]%
  {\PDFParSect{Paragraph}{#1} \originalparagraph[#1]{#2} \endPDFParSect}%
\newcommand{\originalparagraphWithOption}[1]%
  {\PDFParSect{Paragraph}{#1} \originalparagraph{#1} \endPDFParSect}%
```

`subparagraph` Umdefinieren des `\subparagraph`-Befehls

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalsubparagraph\subparagraph%
  \renewcommand{\subparagraph}{%
    \@ifstar{\originalsubparagraphWithStar}%
    {\@ifnextchar[%]
        {\originalsubparagraphWithTwoOption}%
        {\originalsubparagraphWithOption}%
    }%
  }%
}{}
```

Zuordnung der verschiedenen Aufrufvarianten.

```
\newcommand{\originalsubparagraphWithStar}[1]%
  {\PDFParSect{Subparagraph}{#1} \originalsubparagraph*{#1} \endPDFParSect}%
\newcommand{\originalsubparagraphWithTwoOption}[2]%
  {\PDFParSect{Subparagraph}{#1} \originalsubparagraph[#1]{#2} \endPDFParSect}%
\newcommand{\originalsubparagraphWithOption}[1]%
  {\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 `\mnisec`-Befehls

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifundefined{minisec}{}{%
   \let\originalminisec\minisec%
   \renewcommand{\minisec}{%
     \@ifstar{\originalminisecWithStar}%
     {\@ifnextchar[%]
         {\originalminisecWithTwoOption}%
```

```
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

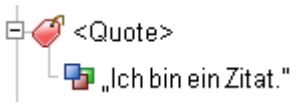


Abbildung 3.2: Struktur einer Zitatumgebung

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 VorgehLen 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 Stukturnames 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

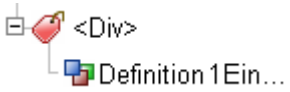


Abbildung 3.4: Struktur eines Theorems

Poteniell 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.

```
  \@ifpackageloaded{theorem}{%
    \newcommand{\@myendtheorem}{%
      \@endtheorem%
      \endPDFSpezialTextObj%
    }%TODO ungetestet
    \let\original@thm\@thm%
    \gdef\@thm#1#2{%
      \PDFSpezialTextObj{Div}%
      \EveryparConfig{H}{true}%
      \PDFMarkContent%
      \global \expandafter \let \csname end#1\endcsname \@myendtheorem%
      \original@thm{#1}{#2}%
    }%
```

Umdefinieren von theroem ohne das theorem-Paket

```
  }{%without theorem-package
    \let\original@begintheorem\@begintheorem%
    \renewcommand{\@begintheorem}{%
      \PDFSpezialTextObj{Div}%
      \EveryparConfig{H}{true}%
      \PDFMarkContent%
      \EveryparConfig{P}{true}%
      \original@begintheorem%
    }%
    \let\original@opargbegintheorem\@opargbegintheorem%
    \renewcommand{\@opargbegintheorem}{%
      \PDFSpezialTextObj{Div}%
      \EveryparConfig{H}{true}%
      \PDFMarkContent%
      \EveryparConfig{P}{true}%
      \original@opargbegintheorem%
    }%
    \let\original@endtheorem\@endtheorem%
    \renewcommand{\@endtheorem}{%
      \original@endtheorem%
      \endPDFSpezialTextObj%
%
    }%
  }%
}{}
```

3.8.4 Aufzählumgebungen

Bei Aufzählungen sieht es im Vergleichzu 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 /Lbl 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

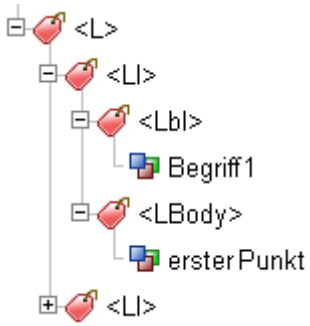


Abbildung 3.5: Struktur einer Liste

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

Variablendeklaration

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

```
\newif\ifItemActive \ItemActivefalse%
\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.

```
\newenvironment{PDFList}{%
  \ifItemActive \closeItem\fi%
  %Liste beginnen
  \addtocounter{ListDepth}{1}%
  %\PDFStructObj{L}{\empty}% Sonst Fehler bei Zugriffsprüfung AA
  \PDFStructObj{L\arabic{ListDepth}}{\empty}%
  %\PDFStructObj{L\romannumeral\theListDepth}{\empty}%
}{%
  \ifItemActive \closeItem\fi%
  %Liste beenden
  \endPDFStructObj%
  \addtocounter{ListDepth}{-1}%
}
```

`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.

```
\newenvironment{PDFListLabel}{%
  \ifItemActive \closeItem\fi%
  \PDFStructObj{LI}{\empty}%
  \global\ItemActivetrue%
  \PDFSpezialTextObj{Lbl}%
  \EveryparConfig{Lbl}{false}%
  \PDFMarkContent%
}{%
  \endPDFMarkContent%
  \endPDFSpezialTextObj%
  \PDFSpezialTextObj{LBody}%
  \EveryparConfig{LBody}{false}%
  %\PDFMarkContent{LBody}% wird über everypar erledigt
}%
```

`\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.

```
\newcommand{\closeItem}{% Altes Item abschließen
  \endPDFMarkContent%
  \endPDFSpezialTextObj%{LBody}
  \endPDFStructObj%
  \global\ItemActivefalse%
}
```

Das eigentliche Umdefinieren

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

`itemize` Umdefinieren der itemize-Umgebung

```
  \let\originalitemize\itemize%
  \let\originalenditemize\enditemize%
  \renewenvironment{itemize}%
    {\begin{PDFList}\originalitemize}%
    {%\ifItemActive \closeItem\fi%
     \originalenditemize\end{PDFList}}%
  %
```

Kennzeichnung der Label für Itemize.

```
\let\originallabelitemi\labelitemi%
\renewcommand{\labelitemi}{%
   \begin{PDFListLabel} \originallabelitemi \end{PDFListLabel}}%
\let\originallabelitemii\labelitemii%
\renewcommand{\labelitemii}{%
   \begin{PDFListLabel} \originallabelitemii \end{PDFListLabel}}%
\let\originallabelitemiii\labelitemiii%
\renewcommand{\labelitemiii}{%
   \begin{PDFListLabel} \originallabelitemiii \end{PDFListLabel}}%
\let\originallabelitemiv\labelitemiv%
\renewcommand{\labelitemiv}{%
   \begin{PDFListLabel} \originallabelitemiv \end{PDFListLabel}}%
%
```

`enumerate` Umdefinieren der enumerate-Umgebung

```
\let\originalenumerate\enumerate%
\let\originalendenumerate\endenumerate%
\renewenvironment{enumerate}%
  {\begin{PDFList}\originalenumerate}%
  {%\ifItemActive \closeItem\fi%
   \originalendenumerate\end{PDFList}}%
%
```

Kennzeichnung der Label für Enumerate.

```
\let\originallabelenumi\labelenumi%
\renewcommand{\labelenumi}{%
  \begin{PDFListLabel} \originallabelenumi \end{PDFListLabel}}%
\let\originallabelenumii\labelenumii%
\renewcommand{\labelenumii}{%
  \begin{PDFListLabel} \originallabelenumii \end{PDFListLabel}}%
\let\originallabelenumiii\labelenumiii%
\renewcommand{\labelenumiii}{%
  \begin{PDFListLabel} \originallabelenumiii \end{PDFListLabel}}%
\let\originallabelenumiv\labelenumiv%
\renewcommand{\labelenumiv}{%
  \begin{PDFListLabel} \originallabelenumiv \end{PDFListLabel}}%
%
```

`description` Umdefinieren der description-Umgebung

```
\let\originaldescription\description%
\let\originalenddescription\enddescription%
\renewenvironment{description}%
  {\begin{PDFList}\originaldescription}%
  {%\ifItemActive \closeItem\fi%
```

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

Kennzeichnung der Label für Description.

```
1154   \let\originaldescriptionlabel\descriptionlabel% aus scrrept
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

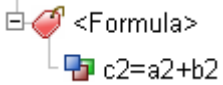


Abbildung 3.6: Struktur einer Formel

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 inaccessibile
1181    \PackageWarning{accessibilty}{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 werde, sollten bei der Definition dynamisch berechnet werden.

Eine Gleiumgebung (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

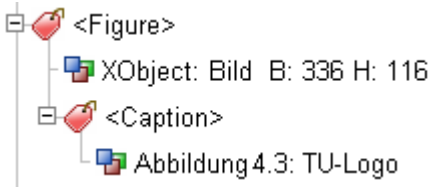


Abbildung 3.7: Struktur einer Grafik

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\original@float\@float%
  \let\originalend@float\end@float%
  \renewenvironment*{@float}[1]{%
    \PDFStructObj{Float}{\csname #1name\endcsname}%
    %\global\numberingparsfalse%
    \original@float{#1}%
  }{%
    \originalend@float%
    \endPDFMarkContent%
    %\global\numberingparstrue%
    \endPDFStructObj%
  }%
}{}
```

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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\original@@makecaption\@makecaption%
 %  \renewcommand{\@makecaption}[3]{%
  \renewcommand{\@makecaption}[2]{%
    \global\numberingparsfalse%
    \PDFSpezialTextObj{Caption}%
    \EveryparConfig{Caption}{false}%
    \PDFMarkContent%
      \PackageWarning{accessibility}{begin makecaption}%
%       \original@@makecaption{#1}{#2}{#3}%
      \original@@makecaption{#1}{#2}%{#3}%
      \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

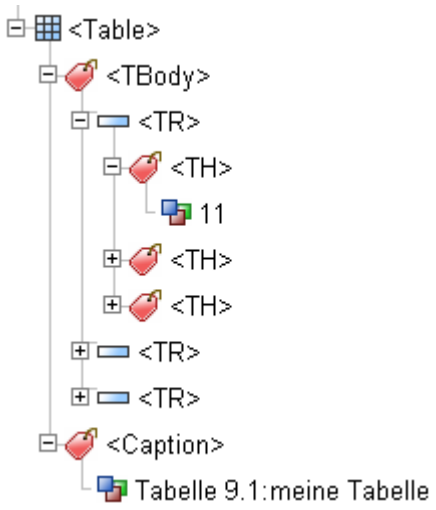


Abbildung 3.8: Struktur einer Tabelle

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.

```
\def\originalamp{&}%
\catcode`\&=\active%
\def&{\originalamp}%

\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originaltabular\tabular%
  \let\originalendtabular\endtabular%
  \renewenvironment*{tabular}{%
    \def&{\endPDFTableCell\originalamp\PDFTableCell}%
    \PDFTable%
    \PDFTableLine%
    \PDFTableCell%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    \originaltabular%
  }{%
    %\pdfliteral{EMC}%
    \def&{\originalamp}%
    \originalendtabular%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    \ifTableCellActive\endPDFTableCell\fi%
    \ifTableLineActive\endPDFTableLine\fi%
    \endPDFTable%
  }%
```

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

```
  \@ifpackageloaded{array}{%
    \let\originalaryend\@arraycr%
    \renewcommand*{\@arraycr}{\endPDFTableCell%
       \endPDFTableLine\PDFTableLine\PDFTableCell\originalaryend}%
  }{% wenn kein anderes Tabellen-Package
    \let\originaltabend\@tabularcr%
    \renewcommand*{\@tabularcr}{\endPDFTableCell%
       \endPDFTableLine\PDFTableLine\PDFTableCell\originaltabend}%
  }%
```

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

```
% \@ifpackageloaded{tabularx}{%
%   \PackageWarning{accessibity}%
%       {The tabularx-package isn't yet fully supported.%
%        You can use the tabular-environemt but not the tabularx.}
% }{}%
% \@ifpackageloaded{longtable}{%
%   \PackageWarning{accessibity}%
%       {The longtable-package isn't yet supported.}
```

```
%   %\tabularnewline  \endhead\endfirsthead\endfoot\endlastfoor
% }{}%
}{}%
```

`tabbing` Umdefinieren der \tabbing-Umgebung.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originaltabbing\tabbing%
  \let\originalendtabbing\endtabbing%
  \renewenvironment*{tabbing}{%
    \PDFTable%
    \let\originalkill\kill%
    \renewcommand{\kill}{\global\AfterKilltrue%
      \originalkill%%
    }%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    \originaltabbing%
  }{%
    \originalendtabbing%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    \endPDFTable%
  }%
  \let\original@startfield\@startfield%
  \renewcommand{\@startfield}{%
    \original@startfield \ifAfterKill\PDFTableCell\fi%
  }%
  \let\original@stopfield\@stopfield%
  \renewcommand{\@stopfield}{%
    \ifAfterKill\endPDFTableCell\fi \original@stopfield%
  }%
  \let\original@startline\@startline%
  \renewcommand{\@startline}{%
    \ifAfterKill\PDFTableLine\fi \original@startline%
  }%
  \let\original@stopline\@stopline%
  \renewcommand{\@stopline}{%
    \original@stopline \ifAfterKill\endPDFTableLine\fi%
  }%
}{}
```

3.9 Elemente auf Zeilenebene

3.9.1 Texthervorhebungen

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 vonn Inhalt und Layout durch den Wegfall der Elemente (<b>,<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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalemph\emph%
  \renewcommand{\emph}[1]{%
    \begin{PDFInlineObjInText}{Span}%
    \originalemph{#1}%
    \end{PDFInlineObjInText}%
  }%
}{}
```

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

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

Wenn das hyperref-Paket geladen ist.

```
  \@ifpackageloaded{hyperref}{%
    \let\original@setref\@setref%
    \renewcommand{\@setref}[3]{%
        \begin{PDFInlineObjInText}{Reference}%
        \original@setref{#1}{#2}{#3}%
        \end{PDFInlineObjInText}}%
    %Linkziele%
    %\let\originalhyper@anchorstart\hyper@anchorstart%
    %\renewcommand{\hyper@anchorstart}%
    %{\pdfliteral{/Span <</E (anchorstart)>> BDC EMC}%
    %\originalhyper@anchorstart}%
    %\let\originalhyper@anchorend\hyper@anchorend%
    %\renewcommand{\hyper@anchorend}{\originalhyper@anchorend
    %\pdfliteral{/Span <</E (anchorend)>> BDC EMC}}%
```

```
    % Einträge im TOC, LOF, LOT
    %\let\originalhyper@linkstart\hyper@linkstart%
    %\renewcommand{\hyper@linkstart}{%
    %    \begin{PDFInlineObjInText}{Reference}%
    %     \originalhyper@linkstart}%
    %\let\originalhyper@linkend\hyper@linkend%
    %\renewcommand{\hyper@linkend}{%
    %     \originalhyper@linkend%
    %    \end{PDFInlineObjInText}}%
    %\useacronym --> Kurzform, Glossarseitezahlen,
    %Indexseitenzahlen, Glossareinträge, Hyperlink
    \let\originalhyperlink\hyperlink%
    \renewcommand*{\hyperlink}[2]{%
        \ifIndexItemActive\else\begin{PDFInlineObjInText}{Reference}\fi%
        %Wenn Index -- folgender Aufruf
        % hyperlink{page.\the\toks@}{\the\toks@}%
        %Bringt Fehler
        \originalhyperlink{#1}{#2}%\relax%
        \ifIndexItemActive\else\end{PDFInlineObjInText}\fi%
    }%
    %href  pdfobleme mit pdf 1.3 \@urlbordercolor nicht definiert
    \let\originalhyper@linkurl\hyper@linkurl%
    \renewcommand{\hyper@linkurl}[2]{%
        \begin{PDFInlineObjInText}{Link}%
        \originalhyper@linkurl{#1}{#2}%
        \end{PDFInlineObjInText}}%
    %
    \let\originalhyper@linkfile\hyper@linkfile%
    \renewcommand{\hyper@linkfile}[3]{%
        \begin{PDFInlineObjInText}{Link}%
        \originalhyper@linkfile{#1}{#2}{#3}%
        \end{PDFInlineObjInText}}%
    %Seitenzahlen in Index, anders da anschließend
    %keine Texterkennung nötig.
    %eigentlich über hyperlink möglich
    \let\originalhyperpage\hyperpage%
    \renewcommand{\hyperpage}[1]{%
        \EveryparConfig{Reference}{true}%
        \PDFMarkContent% kein everypar
        \originalhyperpage{#1}%
        \endPDFMarkContent}%
    % URL
    \let\originalnolinkurl\nolinkurl%
    \renewcommand{\nolinkurl}[1]{%
        \begin{PDFInlineObjInText}{Link}%
        \originalnolinkurl{#1}%
        \end{PDFInlineObjInText}}%
```

Wenn das hyperref-Paket nicht geladen ist.

```
}{% ohne hyperref
```

Umdefinieren des \ref-Befehls

```
      \let\originalref\ref%
      \renewcommand{\ref}[1]{%
        \begin{PDFInlineObjInText}{Reference}%
        \originalref{#1}%
        \end{PDFInlineObjInText}}%
      %
```

Umdefinieren des \pageref-Befehls

```
      \let\originalpageref\pageref%
      \renewcommand{\pageref}[1]{%
        \begin{PDFInlineObjInText}{Reference}%
        \originalpageref{#1}%
        \end{PDFInlineObjInText}}%
  }%
}{}
```

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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalcite\cite%
  \renewcommand{\cite}[2][__empty__]{% #1 Name des Eintages
    \begin{PDFInlineObjInText}{Reference}%
    \ifthenelse{\equal{#1}{__empty__}}%
        {\originalcite{#2}}%
        {\originalcite[#1]{#2}}%
    \end{PDFInlineObjInText}%
  }%
}{}
```

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

3.9.3 eingebettete Objekte im Textfluss

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalverb\verb%
  \renewcommand{\verb}{%
    \begin{PDFInlineObjInText}{Code}%
    \originalverb%
  }%
  \let\originalverb@egroup\verb@egroup%
```

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

`\(\)` An dieser Stelle erfolgt das Umdefinieren 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 verfögbar ist und nicht erst am Seitenende (nach „zig“ Absätzen) vorgelesen wird (vgl. Abbildung 3.10).

Die Latex-Struktur

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

Die PDF-Struktur

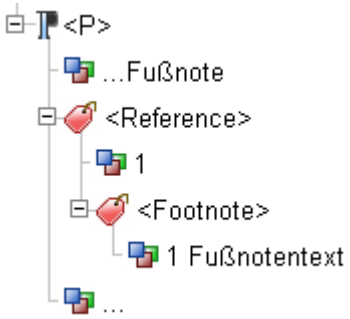


Abbildung 3.10: Fußnotenstruktur im Absatz

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

Variablendeklaration

```
\newcounter{PDFFootnotemark}%
\newcounter{PDFFootnotetext}%
\newcounter{ObjNum}
```

Hilfsmakros

`PDFFootnote` umschließt die gesamte Fußnotenstruktur.

```
\newenvironment{PDFFootnote}{%
  \global\numberingparsfalse%
  \pdfobj reserveobjnum%
  \setcounter{PDFFootnotemark}{\pdflastobj}%
  \pdfobj reserveobjnum%
  \setcounter{PDFFootnotetext}{\pdflastobj}%
}{%
  %\EveryparConfig{\lastEveryparType}{\HelpBool}%
  \global\numberingparstrue%
  \EveryparConfig{\lastEveryparType}{false}%
  \PDFMarkContent%
}
```

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

```
\newenvironment{PDFFootnoteReference}{%
    \xdef\HelpBool{\InlineObj}%
    \EveryparConfig{Reference}{obj}%
    \setcounter{ObjNum}{\theTaggedObj}%
    \PDFMarkContent%
}{%
    \endPDFMarkContent%
    \writeComplexTextObj{\thePDFFootnotemark}%
        {\theObjNum \space \thePDFFootnotetext \space 0 R}%
        {/Reference}{\theTextObjNum}{Page}%
    \xdef\TextArray{\TextArray \theObjHelp\space 0 R \space}%
}
```

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

```
\newenvironment{PDFFootnoteText}{%
   \EveryparConfig{Note}{obj}%
   \setcounter{ObjNum}{\theTaggedObj}%
   \PDFMarkContent%
}{%
   \endPDFMarkContent%
   \writeComplexTextObj%
          {\thePDFFootnotetext}{\theObjNum}%
```

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

Das eigentliche Umdefinieren

Die Befehle stammen aus der soure2e-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

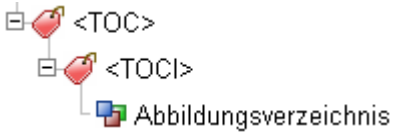


Abbildung 3.11: Struktur eines Inhaltsverzeichnisses

Das eigentliche Umdefinieren

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\original@starttoc\@starttoc%
  \renewcommand{\@starttoc}[1]{%
    \ifthenelse{\equal{#1}{toc}}{% Table of content
        \PDFSpezialTextObj{TOC}\EveryparConfig{TOCI}{true}%
    }{}%
    \ifthenelse{\equal{#1}{lot}}{% List of Tables
        \PDFSpezialTextObj{TOT}\EveryparConfig{TOTI}{true}%
    }{}%
    \ifthenelse{\equal{#1}{lof}}{% List of figures
        \PDFSpezialTextObj{TOF}\EveryparConfig{TOFI}{true}%
    }{}%
    %\ifthenelse{\equal{#1}{brf}}{}}{}% Bibliography
    \original@starttoc{#1}%
    \ifthenelse{\equal{#1}{toc} \or \equal{#1}{lot} \or \equal{#1}{lof}}{%
      \endPDFMarkContent%
      \endPDFSpezialTextObj%
    }{}%
  }%
}{}
```

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalcontentsline\contentsline
  \@ifpackageloaded{hyperref}{%then: Mit hyperref
    \renewcommand{\contentsline}[4]{%
        \originalcontentsline{#1}{#2}{#3\protect\endPDFMarkContent}{#4}%
    }%
  }{%else: ohne Hyperref
    \renewcommand{\contentsline}[3]{%
        \originalcontentsline{#1}{#2}{#3\protect\endPDFMarkContent}%
   }%
  }%
}{}
```

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}
```

```
      \global\BibItemActivefalse%
    \else% erstes Item
      \PDFStructObj{Bibliography}{\empty}%
    \fi%
    \global\BibItemActivetrue%
    \PDFStructObj{BibItem}{\empty}%
    \PDFSpezialTextObj{Lbl}%
    \EveryparConfig{Lbl}{false}%
    \PDFMarkContent%
    \ifthenelse{\equal{#1}{__empty__}}%
        {\originalbibitem{#2}}%
        {\originalbibitem[#1]{#2}}%
    %\endPDFMarkContent% Zu früh, Text wird erst mit everypar gestetzt
    \endPDFSpezialTextObj%
    \PDFSpezialTextObj{LBody}%
    \EveryparConfig{LBody}{false}%
   %\PDFMarkContent{LBody}% wird über everypar erledigt
  }%
}{}
```

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

```
\newif\ifIndexItemActive \IndexItemActivefalse%
```

Das eigentliche Umdefinieren

Umdefinieren der umschließenden `\theindex`-Umgebung.

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originaltheindex\theindex%
  \let\originalendtheindex\endtheindex%
  \renewenvironment{theindex}{%
    \expandafter\originaltheindex\relax%
  }{%
      \endPDFMarkContent%
    \originalendtheindex%
    \ifIndexItemActive%
      \endPDFSpezialTextObj%
      \global\IndexItemActivefalse%
    \fi
    \endPDFStructObj%{Index}%
  }%
}{}
```

Umdefinieren des `\@idxitem`-Befehls.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\original@idxitem\@idxitem%
  \renewcommand*\@idxitem{%
    \ifIndexItemActive% schon welche
      \endPDFMarkContent%
      \endPDFSpezialTextObj%
      \global\IndexItemActivefalse%
    \else% erstes Item
      \PDFStructObj{Index}%
    \fi%
    \global\IndexItemActivetrue%
    \PDFSpezialTextObj{IndexEntry}%
    \EveryparConfig{IndexEntry}{false}%
    \original@idxitem%
  }%
}{}
```

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.

```
\newenvironment*{PDFPageArtefakt}{%
  \pdfliteral{/Artifact <</Type /Pagination>> BDC}%
}{%
  \pdfliteral{EMC}%
}
```

Das eigentliche Umdefinieren

Da Scrpage optimal mit den Klassen des Koma-Scripts zusammenarbeitet, funktioniert es mit scrpage2.

TODO 9 *Funktionstüchtigkeit mit fancyheader und Standardklassen*

```
\ifthenelse{\boolean{@Access@pdf}}{%
   \let\original@thehead\@thehead%
   \renewcommand*{\@thehead}{%
      \ifthenelse{\equal{\original@thehead}{\empty}}{}{%
          \begin{PDFPageArtefakt}%
          \original@thehead%
          \end{PDFPageArtefakt}%
      }%
   }%
   \let\original@thefoot\@thefoot%
   \renewcommand*{\@thefoot}{%
      \ifthenelse{\equal{\original@thefoot}{\empty}}{}{%
          \begin{PDFPageArtefakt}%
          \original@thefoot%
          \end{PDFPageArtefakt}%
       }%
   }%
}{}
```

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.*

```
\newenvironment*{PDFLayoutArtefakt}{%
  \numberingparsfalse%
  \pdfliteral{/Artifact <</Type /Layout>> BDC}%
}{%
  \pdfliteral{EMC}%
  \numberingparstrue%
}
```

Das eigentliche Umdefinieren

Anpassen des `\dotfill`-Befehls.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originaldotfill\dotfill%
  \renewcommand*{\dotfill}{%
    \begin{PDFLayoutArtefakt}%
    \originaldotfill%
    \end{PDFLayoutArtefakt}%
  }%
```

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

```
  \let\originalfootnoterule\footnoterule%
  \renewcommand*\footnoterule{%
    \let\hrule\originalhrule%
    \begin{PDFLayoutArtefakt}%
    \originalfootnoterule%
    \end{PDFLayoutArtefakt}%
    \let\originalhrule\hrule%
  }%
```

Anpassen des `\hrule`-Befehls.

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

```
    \originalhrule#1#2%
    \ifthenelse{\equal{#2}{\z@}}{}{\end{PDFLayoutArtefakt}}%
  }%
```

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.

```
  %vline (2), @arrayrule(2?) greift auf vrule zu
  %Klappt nicht mit Argumentübergabe
  %\let\originalvrule\vrule%
  %\def\vrule#1#2{%
  %  \begin{PDFLayoutArtefakt}%
  %  \originalvrule#1#2%
  %  \end{PDFLayoutArtefakt}%
%  }%
}{}
```

Gepunktete Linien, wie sie im Inhaltsverzeichnis mittels `\dottedtocline` erzeugt werden, werden auch als solches (nämlich „Punkt Punkt ...) vorgelesen. Hierzu wurde die Originaldefinition aus soure2e [BCJ+00] um die pdfliterale ergänzt, wodurch die Linie als Artefakt gekennzeichnet ist und nicht vorgelesen wird.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \def\@dottedtocline#1#2#3#4#5{%
    \ifnum #1>\c@tocdepth \else%
      \vskip \z@ \@plus.2\p@%
      {\leftskip #2\relax \rightskip \@tocrmarg %
      \parfillskip -\rightskip%
      \parindent #2\relax\@afterindenttrue%
      \interlinepenalty\@M%
      \leavevmode%
      \@tempdima #3\relax%
      \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip%
      {#4}\nobreak%
      \begin{PDFLayoutArtefakt}%
      \leaders\hbox{$\m@th \mkern %
        \@dotsep mu\hbox{.}\mkern \@dotsep mu$}\hfill%
      \end{PDFLayoutArtefakt}%
      \nobreak%
      \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
      \par}%
    \fi%
  }%
}{}
```

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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \let\originalmaketitle\maketitle%
  \renewcommand{\maketitle}{%
    \PDFStructObj{Div}{Titlepage}%
    \EveryparConfig{P}{false}%
    %
    \originalmaketitle%
    \endPDFMarkContent%
    \endPDFStructObj%
  }%
}{}%

```

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 TEXfunktionieren mit den gleichen Einschränkungen.

3.13.2 Das graphics-Paket

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

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifpackageloaded{graphicx}{%
    \let\originalincludegraphics\includegraphics%
    \renewcommand{\includegraphics}[2][__empty__]{%
    \global\numberingparsfalse%
     % \PDFInlineObjInText{Figure}%
    \PDFSpezialTextObj{Figure}%
    \EveryparConfig{Figure}{false}%
    \PDFMarkContent%
      \ifthenelse{\equal{#1}{__empty__}}%
          {\originalincludegraphics{#2}}%
          {\originalincludegraphics[#1]{#2}}%
    %  \endPDFInlineObjInText%
```

```
    \endPDFMarkContent%
    \endPDFSpezialTextObj%{Figure}%
    \global\numberingparstrue%
    }%
  }{}%
}{}
```

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 .

```
\ifthenelse{\boolean{@Access@pdf}}{%
    \let\originalpicture\picture%
    \let\originalendpicture\endpicture%
    \renewenvironment{picture}{%
    \global\numberingparsfalse%
    \PDFSpezialTextObj{Figure}%
    \EveryparConfig{Figure}{false}%
    \PDFMarkContent%
    \originalpicture%
 }{%
    \originalendpicture%
    \endPDFMarkContent%
    \endPDFSpezialTextObj%{Figure}%
    \global\numberingparstrue%
    }%
}{}
```

3.13.4 Das babel-Paket

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

```
\newcommand{\convertLanguageInCode}[1]{%
  \gdef\LanguageCode{}%
  \ifthenelse{\equal{#1}{\string danish}}{\gdef\LanguageCode{/Lang(DA)}}{}%
  \ifthenelse{\equal{#1}{\string german}}{\gdef\LanguageCode{/Lang(DE)}}{}%
  \ifthenelse{\equal{#1}{\string ngerman}}{\gdef\LanguageCode{/Lang(DE)}}{}%
  \ifthenelse{\equal{#1}{\string germanb}}{\gdef\LanguageCode{/Lang(DE)}}{}%
  \ifthenelse{\equal{#1}{\string austrian}}{\gdef\LanguageCode{/Lang(DE)}}{}%
  \ifthenelse{\equal{#1}{\string naustrian}}{\gdef\LanguageCode{/Lang(DE)}}{}%
  \ifthenelse{\equal{#1}{\string english}}{\gdef\LanguageCode{/Lang(EN)}}{}%
  \ifthenelse{\equal{#1}{\string USenglish}}{\gdef\LanguageCode{/Lang(EN-US)}}{}%
  \ifthenelse{\equal{#1}{\string american}}{\gdef\LanguageCode{/Lang(EN-US)}}{}%
  \ifthenelse{\equal{#1}{\string UKenglish}}{\gdef\LanguageCode{/Lang(EN-GB)}}{}%
  \ifthenelse{\equal{#1}{\string british}}{\gdef\LanguageCode{/Lang(EN-GB)}}{}%
```

```
  \ifthenelse{\equal{#1}{\string canadian}}{\gdef\LanguageCode{/Lang(EN)}}{}%
  \ifthenelse{\equal{#1}{\string australian}}{\gdef\LanguageCode{/Lang(EN)}}{}%
  \ifthenelse{\equal{#1}{\string newzealand}}{\gdef\LanguageCode{/Lang(EN)}}{}%
  \ifthenelse{\equal{#1}{\string finnish}}{\gdef\LanguageCode{/Lang(FI)}}{}%
  \ifthenelse{\equal{#1}{\string french}}{\gdef\LanguageCode{/Lang(FR)}}{}%
  \ifthenelse{\equal{#1}{\string francais}}{\gdef\LanguageCode{/Lang(FR)}}{}%
  \ifthenelse{\equal{#1}{\string canadien}}{\gdef\LanguageCode{/Lang(FR)}}{}%
  \ifthenelse{\equal{#1}{\string acadian}}{\gdef\LanguageCode{/Lang(FR)}}{}%
  \ifthenelse{\equal{#1}{\string italian}}{\gdef\LanguageCode{/Lang(IT)}}{}%
  \ifthenelse{\equal{#1}{\string norsk}}{\gdef\LanguageCode{/Lang(NO)}}{}%
  \ifthenelse{\equal{#1}{\string nynorsk}}{\gdef\LanguageCode{/Lang(NO)}}{}%
  \ifthenelse{\equal{#1}{\string portuges}}{\gdef\LanguageCode{/Lang(PT)}}{}%
  \ifthenelse{\equal{#1}{\string portuguese}}{\gdef\LanguageCode{/Lang(PT)}}{}%
  \ifthenelse{\equal{#1}{\string brazilian}}{\gdef\LanguageCode{/Lang(PT-BR)}}{}%
  \ifthenelse{\equal{#1}{\string brazil}}{\gdef\LanguageCode{/Lang(PT-BR)}}{}%
  \ifthenelse{\equal{#1}{\string swedish}}{\gdef\LanguageCode{/Lang(SV)}}{}%
  \ifthenelse{\equal{#1}{\string spanish}}{\gdef\LanguageCode{/Lang(ES)}}{}%
   % not surreported in babel:
% Chinese (/Lang{ZH})
% Korean (/Lang{KO}).
  \ifthenelse{\equal{\LanguageCode}{}}{%
% comparing \languagename is tricky. See babel package documentation for more information
\PackageWarning{accessibility}{The chosen language (#1) is not supported %
by Adobe Reader 6.0.}%
}{}%
}
```

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.*

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \AtBeginDocument{%
    \gdef\DocumentLanguage{\languagename}%
    \gdef\ActualLanguage{\languagename}%
    \convertLanguageInCode{\languagename}%
    \pdfcatalog{% Catalog dictionary of PDF output.
      \LanguageCode% Setzt die Sprache
    }%
  }%
}{}
```

Auszeichnung von Sprachwechseln

Hilfsmakro

```
\newcommand{\recognizeLanguageChange}[1]{%
  \ifthenelse{\equal{#1}{\ActualLanguage}}{%
     %keine änderung zu vorher
  }{%
     \gdef\ActualLanguage{#1}%
     \convertLanguageInCode{\languagename}}%
  \ifthenelse{\equal{#1}{\DocumentLanguage}}{%
     \global\LanguageDifffalse%
  }{%
     \global\LanguageDifftrue%
  }%
}
```

`\selectlanguage` `\selectlanguage{Sprache}` vollständige Ersetzung bis zum Dokumentende 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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifpackageloaded{babel}{%
    \let\originalselectlanguage\selectlanguage%
    \renewcommand{\selectlanguage}[1]{%
      \originalselectlanguage{#1}%
      \recognizeLanguageChange{#1}%
    }%
```

`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 gleicht 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.*

```
    \let\originalotherlanguage\otherlanguage%
    \let\originalendotherlanguage\otherlanguage%
    \long\def\otherlanguage#1{%
      \csname selectlanguage \endcsname{#1}%
      \ignorespaces%
      \recognizeLanguageChange{#1}%
    }%
```

```
1833        \long\def\endotherlanguage{%
1834        \originalTeX%
1835        \global\@ignoretrue\ignorespaces%
1836        \recognizeLanguageChange{\languagename}%
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{\languagename}%
1845      }%
1846    }{}%
1847 }{}
```

3.13.5 Das makeidx-Paket

3.13.6 Das glossary-Paket

Glossar

Die Optionen `altlist` und `list` des glossary-Pakets 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.

```
\ifthenelse{\boolean{@Access@pdf}}{%
  \@ifpackageloaded{glossary}{%
    \let\originalnewacronym\newacronym%
    \renewcommand{\newacronym}[4][]{%
  %%%%% Originaldefinition
\ifthenelse{\equal{#1}{}}{\renewcommand\@acrnmcmd{#2}}{%
\renewcommand\@acrnmcmd{#1}}%
    \xdef\expansion{#3}%
\@ifundefined{\@acrnmcmd}{%
\expandafter\newcommand\csname\@acrnmcmd short\endcsname{%
\protect\pdfliteral{/Span <</E (\expansion)>> BDC}%
#2%
\protect\pdfliteral{EMC}%
\protect\glsxspace}%
\expandafter\newcommand\csname\@acrnmcmd @nx@short\endcsname{%
\protect\pdfliteral{/Span <</E (\expansion)>> BDC}%
#2%
\protect\pdfliteral{EMC}}%
\expandafter\newcommand\csname\@acrnmcmd long\endcsname{%
#3\protect\glsxspace}
\expandafter\newcommand\csname\@acrnmcmd @nx@long\endcsname{#3}
\def\@acrn@entry{#4}%
{%
\expandafter\@gls@getdescr\expandafter{\@acrn@entry}%
\let\glodesc\@glo@desc%
\def\glolong{#3}%
\@onelevel@sanitize\glolong
\def\gloshort{\noexpand\acronymfont{#2}}%
\@onelevel@sanitize\gloshort
\expandafter\protected@xdef\expandafter\@acrnamefmt{\@acronymnamefmt}
\expandafter\protected@xdef\expandafter\@acrdesc{\@acronymdescfmt}
}%
\@acr@addtolist{\@acrnmcmd}
\@glo@tb=\expandafter{\@acrn@entry}%
\protected@edef\@acr@glsentry{name={\@acrnamefmt},%
format=glsnumformat,sort={\@acrnmcmd},\the\@glo@tb,%
description={\@acrdesc}}%
\@glo@tb=\expandafter{\@acr@glsentry}%
\newboolean{\@acrnmcmd first}\setboolean{\@acrnmcmd first}{true}
\expandafter\protected@edef\csname \@acrnmcmd\endcsname{%
\noexpand\@ifstar{\csname @s@\@acrnmcmd\endcsname}{%
\csname @\@acrnmcmd\endcsname}}
```

```
\ifglshyperacronym % hyperlinks
\expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})\noexpand\unsetacronym{\@acrnmcmd}%
}{\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
\expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})%
\noexpand\unsetacronym{\@acrnmcmd}}{%
\noexpand\xacronym{\the\@glo@tb}{%
\noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
\else % no hyperlinks
\expandafter\protected@edef\csname @\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})\noexpand\unsetacronym{\@acrnmcmd}%
}{\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}%
\noexpand\glsxspace}
\expandafter\protected@edef\csname @s@\@acrnmcmd\endcsname{%
\noexpand\ifthenelse{\noexpand\boolean{\@acrnmcmd first}}{%
\noexpand\expandafter
\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@long\endcsname\noexpand\@acrnmins\
(\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\csname\@acrnmcmd @nx@short\endcsname}%
})%
\noexpand\unsetacronym{\@acrnmcmd}}{%
\noexpand\acronym{\the\@glo@tb}{%
\noexpand\acronymfont{\noexpand\expandafter\noexpand\MakeUppercase
\csname\@acrnmcmd @nx@short\endcsname}%
\noexpand\@acrnmins}}\noexpand\glsxspace}
\fi
}{%
\PackageError{glossary}{Command '\expandafter\string
\csname\@acrnmcmd\endcsname' already defined}{%
The command name specified by \string\newacronym already exists.}}}
  %%%%% Originaldefinition
```

```
1942   }{}%
1943 }{}
```

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-LaTeX-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-LaTeX-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-LaTeX-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 LaTeX 2ε 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 LaTeX-Dokumenten hoher Accessibility*. Technische Universität Dresden, Fakultät Informatik, Version 1 Auflage, April 2007.