Anfertigen von hochwertigen Tabellen mit LaTeX*

Simon Fear
300A route de Meyrin Meyrin
Switzerland.

Überführt ins Deutsche
von Thomas Manderla und
Christine Römer†

Printed 30. März 2011

Zusammenfassung

Dieser Artikel beschreibt einige zusätzliche Befehle in `booktabs`, geschaffen für die Verbesserung der Qualität von Tabellen in LaTeX. Richtlinien, was eine gute Tabelle ausmacht, werden in diesem Zusammenhang auch angegeben. Die Ausführungen sind zum `booktabs`-Paket aus dem Jahr 2000 (Version 1.61), das der Ausgabe von 1995 (Version 1.00) einige Verbesserungen hinzugefügt hatte, vor allem `longtable`-Kompatibilität. Weitere Ausgaben (Versionen 1.618, 1.6180 und 1.61803) sind bloß *bug patches* und Unterstützung für das `colortbl`-Paket[1].

1 Einführung

Die unten beschriebenen Routinen sollten das einfache Erstellen von Tabellen, wie sie in wissenschaftlichen Büchern und Zeitschriften erscheinen, ermöglichen. Was sie von gewöhnlichen LaTeX-Tabellen unterscheidet, ist die vorgegebene Nutzung von Abstand über- und unterhalb von Linien, sowie Linien unterschiedlicher 'Dicke'. Was sie weiterhin von Tabellen, wie sie von vielen Leuten, die LaTeX nutzen, erstellt werden, unterscheidet, ist das Fehlen vertikaler und doppelter Linien.

Es muss eine deutliche Unterscheidung zwischen dem, was man hier eine *formale Tabelle* nennt, was ein Set an Werten in beschrifteten Spalten ist, und dem, was

*Diese Datei basiert auf der booktabs-Version vl.61803 und der Revision von 2005/04/14.

†christine_roemer@t-online.de

[1]Von Danie Els (`dnjels@sun.ac.za`) in Abwesenheit des Autors.

man *Tableau* nennen werde, vornehmen. Letzteres ist das, was im LaTeX-Handbuch dargestellt ist, und zunehmend auch als Ausgabe vieler Datenbankverwaltungssysteme zu finden ist. Solche Tableaus haben wahrscheinlich Icons im Überfluss und verwenden sicherlich auch Farben. Das Layout solche eines em Tableaus ist (hoffentlich) etwas Einmaliges, wenn man das Durcheinander an Material, das der Designer zu einer sinnvollen Kombination zusammenzustellen versucht, bedenkt. Das Layout einer *Tabelle* hingegen hat sich durch Jahrhunderte an Erfahrung etabliert und sollte nur in außergewöhnlichen Umständen geändert werden.

Zur Illustration kann dieses Tableau aus dem LaTeX-Handbuch herangezogen werden (S. 64 alte Edition):

Mücken	Gramm	$13.65
	je	.01
Gnu	ausgestopft	92.50
Emu		33.33
Gürteltier	gefroren	8.99

Das ist ein Mischmasch an Informationen, das zwar wahrscheinlich, so wie es ist, vernünftig dargestellt ist (Aber ist der Emu nun ausgestopft oder nicht?). Jedoch sollte es als veröffentlichte Tabelle viel eher gemäß der Richtlinien, die weiter unten auf der Seite im Handbuch empfohlen werden, erscheinen:

Artikel		
Tier	Beschreibung	Preis ($)
Mücke	pro Gramm	13.65
	pro Stück	0.01
Gnu	ausgestopft	92.50
Emu	ausgestopft	33.33
Gürtetier	gefroren	8.99

Es bedarf viel weniger Mühe, das als formale Tabelle zu setzen. Es muss kein neues Layout für alles, was gemacht wird, ausgearbeitet werden. Zudem kann fast gewiss sein, dass die Daten nicht falsch gelesen werden, weil der Leser nicht lernen muss, wie er irgendeine neuartige Präsentation zu lesen hat.

Unglücklicherweise kann die obige Tabelle nicht in reinem LaTeX, erzeugt werden. Sie kann so gesetzt werden, aber trotz bester Bemühungen erzeugt das Nutzen simpler `\hline`-Befehle das Nachfolgende:

Artikel		
Tier	Beschreibung	Preis ($)
Mücke	pro Gramm	13.65
	pro Stück	0.01
Gnu	ausgestopft	92.50
Emu	ausgestopft	33.33
Gürteltier	gefroren	8.99

Zu beachten ist (wenn es nicht schon offensichtlich ist), dass nicht genug Abstand zwischen der obersten Linie und dem großen 'A' von 'Artikel' ist. Ähnliches gilt für die anderen Linien: Ein direkter Vergleich zeigt dies deutlich. Außerdem sind in der ersten Version die oberen und unteren Linien stärker als die mittlere, die wiederum stärker als die Teillinie unterhalb von 'Artikel' ist. Natürlich *könnte* man `\doublerulesep` neu definieren und dann `\hline\hline` nutzen, um einen ähnlichen Effekt zu erzielen. Und man könnte *struts* nutzen, um die Abstände zu verbessern. Aber es ist nicht nötig, an solche Sachen denken zu müssen. Der `booktabs`-Style definiert seine Befehle so, dass sich um solche Dinge automatisch gekümmert wird.

Allgemein würde ich sagen, dass dieses Paket von keinem Interesse für diejenigen ist, die nach einer Alternative zu PicTEX suchen, um schicke Tableaus zu zaubern. Eher ist es eine Gestaltungsrichtlinie im Bereich Tabellenlayout für Autoren von wissenschaftlichen Artikeln und Büchern. Es geht nicht zu weit, zu sagen, dass, wenn Sie mit den Befehlen dieses Pakets keine Tabelle erstellen können, es neu entwerfen sollten.

1.1 Eine Anmerkung zur Terminologie

Im Britischen Setzen wird eine 'line' stets 'rule' genannt. Vielleicht verwirrenderweise (jedenfalls aus historischen Gründen) wird die 'thickness' einer Linie oft als 'width' bezeichnet. (wobei nahezu jeder andere dies 'depth' oder 'height' nennen würde, wenn sie sich auf eine horizontale Linie bezieht). Eine 'thick black line' wird 'heavy rule' genannt. Ich habe diese Terminologie in den meisten der neuen Befehle beibehalten. Wenn es für nichts anderes gut ist, dann zumindest, um Verwirrung mit `\hline` auszuschließen. In der Deutschen Übersetzung wird an der Stelle von 'Stärke' gesprochen.

2 Das Layout formaler Tabellen

Man geht nicht sehr weit fehl, wenn man zwei einfache Regeln im Hinterkopf behält:

1. Nie vertikale Linien benutzen.
2. Nie doppelte Linien benutzen.

Diese Richtlinien wirken extrem, aber mir ist noch kein gutes Argument, warum man sie brechen sollte, untergekommen. Zum Beispiel, wenn man das Gefühl hat, dass die Information in der linken Hälfte der Tabelle von der rechten Seite so verschieden ist, dass sie durch eine vertikale Linie getrennt werden muss, so sollte man stattdessen lieber zwei Tabellen verwenden. Nicht jeder hält sich an die zweite Regel: Ich habe für einen Verleger gearbeitet, der auf doppelte Linien über Summen bestand. Meine Wahl wäre das nicht gewesen.

Es gibt drei weitere Regeln, die es wert sind, hier erwähnt zu werden, weil sie normalerweise außerhalb der Kreise von professionellen Setzern und Redakteuren unbekannt sind:

3. Stückzahlen sollten in den Spaltenkopf (nicht in den Körper der Tabelle) gesetzt werden.
4. Einem Dezimalpunkt sollte stets eine Ziffer vorangehen; also 0.1 *nicht* bloß.1.
5. 'Ditozeichen', wie Gänsef"ußchen, sollten nicht benutzt werden, um einen Wert zu wiederholen. Unter vielen Umständen macht eine freie Zelle das Ganze genauso gut. Falls nicht, sollte der Wert wiederholt werden.

Ob Sie den kleinen Penibilitäten folgen wollen oder nicht, wenn Sie nur die folgenden Befehle in Ihren formalen Tabellen benutzen, werden Ihnen Ihre Leser dankbar sein. Ich möchte betonen, dass die Richtlinien nicht nur dazu da sind, die Pedanten glücklich zu machen. Das Prinzip ist, dass eine verstärkte Struktur in der Präsentation bereits von Beginn an die Gedanken strukturiert.

3 Verwenden der neuen Befehle

`\toprule` `\midrule` `\bottomrule`

Im einfachsten aller Fälle beginnt eine Tabelle mit `\toprule`, hat eine einzelne Reihe von Spaltenköpfen und dann eine trennende Linie, die `\midrule`. Nach den Datenspalten wird mit `\bottomrule` abgeschlossen. Die meisten Buchverleger setzen `\toprule` und `\bottomrule` stärker (also dicker, siehe Punkt 1.11.1) als die zwischenliegende `\midrule`. Wenn die Tabellen jedoch in sehr kleinen Schriftgrößen erscheinen, ist es manchmal unmöglich, diese Unterscheidung zu treffen und zudem setzen manche Journals routinemäßig alle Linien in der gleichen Stärke.

Die Linienbefehle hier verwenden alle eine Voreinstellung, der im Dokument (vorzugs- aber nicht notwendigerweise in der Präambel) nachjustiert werden kann. Für die oberste und die unterste Linie ist diese Voreinstellung `\heavyrulewidth` und für mittlere Linien ist er `\lightrulewidth` (eine genauere Beschreibung folgt). In sehr seltenen Fällen, wenn etwas besonderes getan werden muss, können die optionalen Argumente in die Linienbefehle eingesetzt werden, welche folgende formale Syntax haben:

`\toprule[`⟨*wd*⟩`]`
`\midrule[`⟨*wd*⟩`]`
`\bottomrule[`⟨*wd*⟩`]`

Hierbei ist ⟨*wd*⟩ eine TEXdimension (zum Beispiel 1pt, .5em, etc.).

Alle hier beschriebenen Linienbefehle stehen nach dem Abschluss `\\` der vorigen Reihe (außer `\toprule`, der gleich nach dem `\tabular{}`befehl kommt). In anderen Worten, sie stehen genau da, wo simples LATEX `\hline` oder `\cline` erlaubt.

`\cmidrule` Häufig wird eine Teillinie benötigt, die sich nur über einige der Spalten erstreckt, wofür ein `\cmidrule` (Der Pendant vom LaTeX`\cline` Befehl) benötigt wird. Für gewöhnlich sollte diese Linie nicht über die gesamte Breite der Spalten reichen. Dies ist insbesondere der Fall, wenn ein `\cmidrule` direkt nach dem Ende eines anderen begonnen werden muss (LaTeX`\cline`n stoßen hier zusammen, wenn man nicht ganz besonders vorsichtig mit `\extracolsep` ist). Von daher wird man für gewöhnlich die optionalen 'Trimm'befehle nutzen wollen.

Die Trimmbefehle stehen, wenn sie überhaupt genutzt werden, in Klammern (wie diese hier), ohne trennende Leerzeichen. Die möglichen Festlegungen sind `r`, `r{`⟨*wd*⟩`}`, `l` und `l{`⟨*wd*⟩`}`, oder jede Kombination davon, wobei ⟨*wd*⟩ eine Abmessung ist, und `r` und `l` anzeigen, ob das rechte und/oder linke Ende der Linie beschnitten werden soll. Die Form ohne explizites Argument entspricht `r{\cmidrulekern}`, wobei `\cmidrulekern` voreingestellt 0.5 em enthält, aber es kann vom Nutzer in der Präambel festgelegt werden.[2]

Hier ist ein erläuterndes Beispiel: `(lr{.75em})` macht einen vorgegebenes linkes Beschneiden und genau 0.75 em auf der rechten Seite. `(r{.75em}l)` ist hier genauso gültig.[3]

Die vollständige Syntax des Befehls lautet

`\cmidrule[`⟨*wd*⟩`](`⟨*trim*⟩`){`a–b`}`

wobei ⟨*wd*⟩ ein optionaler Linienstärkenbefehl ist, in eckigen Klammern [wie diese hier] (die Voreinstellung hierbei ist `\cmidrulewidth`) und das letzte Argument, *das nicht optional ist*, gibt die Zahlen der Spalten an, die überspannt werden sollen.

Als Beispiel einer Anwendung dieser Befehle kann der Code, der die Beispieltabelle oben erzeugte, dienen:

```
\begin{tabular}{@{}llr@{}} \toprule
\multicolumn{2}{c}{Artikel} \\ \cmidrule(r){1-2}
Tier  & Beschreibung & Preis (\$)\\ \midrule
Mücke & pro Gramm    & 13.65 \\
      & pro Stück    & 0.01 \\
Gnu   & ausgestopft  & 92.50 \\
Emu   & ausgestopft  & 33.33 \\
Gürteltier & gefroren & 8.99 \\ \bottomrule
\end{tabular}
```

`\addlinespace` Gelegentlich möchte man ein wenig zusätzlichen Abstand zwischen manchen Reihen der Tabelle einzufügen; zum Beispiel vor der letzten Reihe, wenn sie eine Summe angibt. Dazu bedarf es bloß des Einfügens von

[2] Nutzerrückmeldungen ergaben, dass die Vorgabe der Version 1.00, 0.25 em, zu klein war. Entschuldigung für alle Verluste an Rückwärtskompatibilität. Zu beachten ist, dass das ursprüngliche Verhalten einfach durch `\cmidrulekern` in der Präambel wiederhergestellt werden kann, oder einfach `(r{.25 em})` verwendet werden kann.

[3] `(lrrlr{.75em})` macht übrigens das gleiche: Nur die zuletzt angetroffenen linken und rechten Festlegungen werden angewandt.

`\addlinespace[`⟨*wd*⟩`]`

nach den \\ Abschlusszeichen. Zwischen normalen Textreihen ist der Effekt genau wie bei herkömmlichem LaTeXverwendung von `\\[\defaultaddspace]`, was ich eher klobig finde. Außerdem ist es besser als `\\ \\`, was zu viel Abstand einfügt. Zudem kann `\addlinespace` vor, nach oder zwischen Zeilen verwendet werden, wenn man das genaue Ausmaß an Abstand, der eingefügt werden soll, kontrollieren will. Der voreingestellte Abstand vor oder nach einer angrenzenden Linie wird entweder durch genau `\defaultaddspace` oder die Größe des Abstands, die im optionalen Argument angegeben wird, ersetzt.[4]

4 Missbrauch der neuen Befehle

Für ein Zusammenarbeiten der neuen Linienbefehle mit `\hline` oder `\cline` kann nicht garantiert werden, auch wenn diese verfügbar und unverändert bleiben. Man kann sich aber auch keinen Grund vorstellen, warum man sie vermischen wollte.

Viel wichtiger ist, dass die Linien, die durch neue Befehle erstellt werden, unter keiner Garantie mit Vertikalen verbunden werden, die durch `{|}` -Buchstaben in der Präambel gebaut werden. Das ist ein Feature (siehe oben). Man sollte in Tabellen einfach keine vertikalen Linien verwenden, Punkt.

`\morecmidrules` Wenn man sich vom Benutzen einer Doppellinie beim besten Willen nicht abhalten kann, wird selbst eine so bizarre Konstruktion wie `\toprule\bottomrule\midrule` ohne eine Fehlermeldung funktionieren (genauso wie man `\hline` verdoppeln kann). Diese Linien werden vom normalen LaTeXseparator `\doublerulesep` getrennt. Wenn die Perversion, doppelte Linien zu setzen, soweit geht, doppelte `\cmidrules` einzubauen, so wird man den zusätzlichen Befehl `\morecmidrules` benötigen, um das ordentlich zu machen, denn normalerweise sind zwei `\cmidrules` in einer Reihe eine vernünftige Konstruktion, die nach zwei Linien in derselben 'Linienreihe' ruft. Daher schreibt der zweite Befehl bei

```
\cmidrule{1-2}\cmidrule{1-2}
```

eine Linie, die die erste genau überschreibt. Ich nehme an, man möchte dann eher

```
\cmidrule{1-2}\morecmidrules\cmidrule{1-2}
```

was eine doppelte Linie zwischen den Spalten eins und zwei ergibt und durch `\cmidrulesep` getrennt wird (Hinweis: Da eine `\cmidrule` normalerweise sehr dünn ist, richtet das normale `\doublerulesep` hier wahrscheinlich zu viel Abstand ein). Eine ganze neue Reihe an Linien sollte abgeschlossen werden, bevor der

[4] Das ist eine Änderung zur Version 1.00, wo der Abstand manchmal *zusätzlich* zum voreingestellten Abstand um die Linie berechnet wurde.

Befehl \morecmidrules eingegeben wird. Zu beachten ist, dass \morecmidrules keinerlei Effekt hat, wenn er nicht unmittelbar einer \cmidrule folgt. (d. h. es ist kein allgemeiner abstandschaffender Befehl).

\specialrule Falls man das außergewöhnliche Bedürfnis hat, genau 0.5 em, zwischen zwei Linien festzulegen, so ließe sich eine Konstruktion wie \midrule \addlinespace[.5em] \midrule verwenden. In einem seltenen Anfall von Toleranz habe ich dann doch den Befehl

\specialrule{⟨*wd*⟩}{⟨*abovespace*⟩}{⟨*belowspace*⟩}

erstellt, bei dem alle drei Argumente obligatorisch sind. Aber, wer den häufig verwendet, hat Zweck und Inhalt der obigen Richtlinien falsch verstanden. Eine vorangehende Linie fügt ihren voreingestellten Abstand nicht unter sich und eine nachstehende Linie fügt keinen Abstand über sich hinzu, folglich bekommt man *genau* den Abstand, der in den Argumenten angegeben wird.[5]

5 Booktabs und longtables

Wenn man sowohl das booktabs- als auch das longtable-Paket geladen hat, können die booktabs Linienbefehle ganz genau wie oben beschrieben in einer longtable verwendet werden.

Es gibt eine Neuheit, die es wert ist, erwähnt zu werden: Innerhalb einer longtable können die optionalen linken und rechten Trimmbefehle verwendet werden, die normalerweise nur für \cmidrules, mit \toprule, \midrule und \bottomrule (und wenn es sein muss auch mit\specialrule) funktionieren. Nutzer, die die vorherige Ausgabe wegen longtable-Kompatibilität gehackt haben,[6] schienen alle Linien auf der rechten Seite um 0.5 em beschneiden zu wollen. Ich denke, dasselbe lässt sich durch @{} als Spezifikation der letzten Spalte erreichen. Dennoch, nach dem Ausarbeiten des restlichen Codes war es ein leichtes, Parsing für die optionalen Argumente hinzuzufügen, also hab ich es gemacht. (Ich habe es aber nicht durchgezogen, noch das optionale Beschneiden der Linien *außerhalb* einer longtable zu ermöglichen. Das wäre eine Riesenarbeit gewesen. Wenn man unbedingt beschnittene Linien haben muss, dann sollte man longtables verwenden!)

Ein Hinweis etwas technischer Natur: Innerhalb einer longtable ergeben \hline und \hline\hline beide eine *doppelte* Linie (um Seitenumbrüche an dieser Stelle zu ermöglichen). Die booktabs-Linie tut das hingegen *nicht*. Die automatische Doppelung von \hline in der longtable ist fragwürdig, sogar nach Aussage der Dokumentation zu dem Paket. Aber doppelte booktabsLinien ergeben auch beinahe gar keinen Sinn. Im unglücklichen Fall, dass eine booktabslinie an einem Seitenumbruch erscheinen sollte, muss man die notwendigen Anpassungen per Hand machen. (Im Allgemeinen heißt das, die störende Linie zu löschen.)

[5] Das ist eine Änderung zur Version 1.00, die vielmehr einen extra \doublerulesep, wo immer sie konnte, hinzufügte.

[6] Jim Service war der Erste.

6 Booktabs und das colortbl-Paket

`Booktabs` ist jetzt mit dem `colortbl`-Paket kompatibel.[7] Der Befehl `\arrayrulecolor` wird farbige Linien hervorrufen, wenn das `colortbl`-Paket geladen ist.

7 Technische Zusammenfassung der Befehle

Die neuen Linienbefehle sind innerhalb der voreingestellten `tabular`- (und `array`-) Umgebungen gültig, im modifizierten `tabular` und `array` von `\usepackage{array}`, und innerhalb der voreingestellten Tabellen und `longtables` nach `\usepackage{longtable}`.

Die Befehle folgen der Standardplatzierungssyntax von `\hline`. Es kann Platz (sogar ein, aber nicht zwei Enter) zwischen aufeinanderfolgenden Linienbefehlen sein. [8]

Was sich als große Veränderung zu vorherigen Ausgaben zeigt, ist, dass ich innerhalb des Macrocodes jetzt drei Linienklassen definiere. (Im normalen Gebrauch benötigt man diese Definitionen nicht, deshalb habe ich sie oben nicht erwähnt. Eine Linie der Klasse 1 (ansonsten eine 'normale' Linie genannt) ist jede `\toprule`, `\midrule`, `\bottomrule`, oder `\cmidrule`. Die Linien der Klasse 2 sind `\specialrule` und `\addlinespace`. Schließlich ist eine Linie der Klasse 0 keine der vorhergehenden – oder in anderen Worten, überhaupt keine Linie.[9] Zu beachten ist, dass `\addlinespace` als eine Klasse-2-Regel gilt, nicht als Klasse-0-Text.

Im Folgenden, wird jeder Befehl im 'normalen Gebrauch', beschrieben, also die Linie wird zwischen zwei Textzeilen verwendet. (Oder technischer, ihr geht eine Klasse-0-Linie vorher und eine schließt sich ihr an). Danach wird ein Blick auf die Ausnahmen geworfen.

`\toprule[`⟨*wd*⟩`]`

Eine Linie von der Stärke ⟨*wd*⟩ (default `\heavyrulewidth`) mit `\abovetopsep` Abstand darüber und `\belowrulesep` zusätzlichen vertikalen darunter eingefügten Abstand. Als Voreinstellung ist `\abovetopsep` null, was für eine Linie, die ganz oben stehen soll, recht vernünftig scheint. Wenn die Tabellen jedoch Beschriftungen haben, kann es sinnvoller sein, `\abovetopsep` zu verwenden, um eine angemessene Menge an Abstand zwischen Beschriftung und Tabelle einfügen, als im Eifer des Gefechts daran zu denken, einen `\vspace{}`-Befehl einzusetzen.

`\midrule[`⟨*wd*⟩`]`

[7] Seit v1.6180

[8] Eine willkommene Veränderung gegenüber Version 1.00, bei der Abstand zwischen Linienbefehlen für rätselhafte Fehlermeldungen sorgte.

[9] Mit der Ausnahme, dass `\hline` und `\cline` Klasse 0 sind. Doch es gibt keinen Grund, darüber schlaflose Nächte zu verbringen, da man die beiden Linienziehsysteme nicht durcheinander bringen möchte.

Eine ⟨*wd*⟩ (default `\lightrulewidth`) Linie mit `\aboverulesep` Abstand darüber und mit `\belowrulesep` Abstand darunter.

`\bottomrule[`⟨*wd*⟩`]`

Eine ⟨*wd*⟩ (default `\heavyrulewidth`) Linie mit `\aboverulesep` Abstand darüber und mit `\belowbottomsep` Abstand darunter. Voreingestellt ist `\belowbottomsep` null[10] war. Es gibt einen häufig vorkommenden und legitimen Grund, warum man Platz unter der untersten Linie schaffen möchte: nämlich, wenn es um Fußnoten in Tabellen geht. Wenn man nicht die Voreinstellung überschreibt, kann man `\bottomrule \addlinespace[\belowrulesep]` verwenden oder einen passend gestalteten *strut* in den Fußnotentext setzen. Aber damit es sich sinnvoll in einer `longtable`-Fußzeile verhält, muss die Voreinstellung null sein.

`\cmidrule[`⟨*wd*⟩`](`⟨*trim*⟩`){`a–b`}`

Eine ⟨*wd*⟩ (default `\cmidrulewidth`) Linie mit `\aboverulesep` Abstand darüber (außer sie folgt einer anderen `\cmidrule`, in welchem Fall sie der gleichen vertikalen Anordnung folgt. Das gleiche passiert, wenn sie auf eine `\morecmidrules` folgt, getrennt von einer vorhergehenden `\cmidrule` um `\cmidrulesep`). Eine `\cmidrule` hat `\belowrulesep` unter sich (außer ihr folgt eine andere `\cmidrule`, in welchem Fall die folgende Linie der gleichen vertikalen Anordnung folgt; oder wenn nach ihr eine `\morecmdirules` kommt, in welchem Fall `\cmidrulesep` unter ihr Platz gelassen wird).

Die `\cmidrule` überspannt Spalten a bis b genauso, wie es im obligatorischen Argument angegeben wurde. Das optionale Argument ⟨*trim*⟩, das wenn überhaupt in Klammern steht, kann jede Sequenz der Zeichen `r`, `l` und `{`⟨*wd*⟩`}` beinhalten, wobei letzteres das Kerning auf die Seite anwendet, die das direkt vorhergehende Zeichen angab. (Momentan wird hier noch nicht auf Fehler überprüft, es gilt also darauf zu achten, die Syntax richtig zu setzen.)

`\morecmidrules`

Instruiert LaTeX damit, eine neue Reihe von `\cmidrules` zu beginnen, die von der letzten durch `\cmidrulesep` getrennt wird. Hat in einem anderen Kontext keine Bedeutung.

`\specialrule{`⟨*wd*⟩`}{`⟨*abovespace*⟩`}{`⟨*belowspace*⟩`}`

Eine ⟨*wd*⟩ Linie – zu beachten ist, dass das hier ein obligatorisches Argument ist – mit ⟨*abovespace*⟩ darüber und ⟨*belowspace*⟩ darunter.

`\addlinespace[`⟨*wd*⟩`]`

[10] Das ist eine Veränderung zur Version 1.00, bei der stets ein `\belowrulesep`

Technisch gesehen hat das denselben Effekt wie `\specialrule{0pt}{0pt}{`⟨*wd*⟩`}`, d. h. eine Linie ohne Stärke und ohne Abstand darüber und mit ⟨*wd*⟩ (default `\defaultaddspace`) Abstand darunter. Dieser Befehl wurde in erster Linie designed , um im Tabellenkörper Abstände hinzuzufügen, kann aber auch genutzt werden, um eine genaues Maß an Abstand über oder unter einer Klasse-1-Linie festzulegen.

Nun zu den Ausnahmen zu den oberen Befehlen. Es wurde bereits in den Definitionen gezeigt, dass vor und nach den Klasse-2-Linien genauso viel Abstand gelassen wird, wie in den Argumenten angegeben. Das hat zur Folge, dass eine Klasse-2-Linie den Abstand, der normalerweise durch eine Klasse-1-Linie erzeugt würde, unterdrückt (Z. B. `\belowrulesep` nach einer `\toprule`) und ihn mit dem Argument der Klasse-2-Linie ersetzt. Analog dazu wird bei der Kombination {Klasse-2-Linie}{Klasse-1-Linie} der übliche Abstand oberhalb der Klasse-1-Linie (z. B. `\aboverulesep`) unterdrückt. Allerdings wird bei der Kombination {Klasse-2-Linie}{Klasse-2-Linie} kein Abstand unterdrückt: die Linien werden sowohl durch {⟨*belowspace*⟩} der ersten Linie als auch durch {⟨*abovespace*⟩} der zweiten Linie getrennt. Nicht zuletzt ergibt die Kombination {Klasse-1-Linie}{Klasse-1-Linie} immer Linien, die durch `\doublerulesep` getrennt sind und sie unterdrückt allen normalen Abstand, der zwischen den Linien erzeugt worden ist (aber lässt über der ersten und unter der zweiten normale Abstände).

Als eine Ausnahme zu dieser Ausnahme schließt 'Klasse-1-Linie' `\cmidrule` aus. Derartige Linien lassen sich mit anderen `\cmidrules` und `\morecmidrules` ganz normal wie oben beschrieben kombinieren. Ich weiß nicht, und es ist mir auch egal, was die Kombination `\toprule\cmidrule{1-2}\midrule` ergäbe. Ich sehe keine Entschuldigung für eine derartige Anwendung.

Die voreingestellten Ausmaße werden zu Beginn der Makrobeschreibung (Section 9) definiert. Der Nutzer kann diese Voreinstellungen in der Präambel ändern oder auch außerhalb einer Tabellenumgebung, ganz einfach, indem er einen Befehl in genau demselben Format wie in Sektion 9 einfügt. Eine Neudefinition wird bis zum Ende des Dokuments, oder bis sie wieder neu definiert wird, wirken. *Innerhalb einer Tabelle* müsste man die Änderung global in einer `noalign`-Gruppe machen: z. B. `\noalign\{\global\abovetopsep=1em\toprule}`. Ich hoffe, wird nie nötig sein.

8 Danksagungen

Bin natürlich dem DEK und Lamport riesig zu Dank verpflichtet; das optionale Argument und das `\cmidrule`zeug insbesondere wurde von `latex.sty` geklaut. Das Dokumentationtreiberzeug wurde von der tools-Paketbeschreibung `dcolumn.dtx` von David Carlisle geklaut.

Für das beta-Testen und die Unterstützung ...

9 Der code

Die aktuelle Version wird am Anfang der Datei definiert, was in etwa so aussieht

```
⟨*package⟩
%\NeedsTeXFormat{LaTeX2e}
%\ProvidesPackage{booktabs}
%        [\filedate\space version\fileversion]
```

Zuerst werden die oben beschriebenen neuen Dimensionen eingerichtet:

```
\newdimen\heavyrulewidth
\newdimen\lightrulewidth
\newdimen\cmidrulewidth
\newdimen\belowrulesep
\newdimen\belowbottomsep
\newdimen\aboverulesep
\newdimen\abovetopsep
\newdimen\cmidrulesep
\newdimen\cmidrulekern
\newdimen\defaultaddspace
\heavyrulewidth=.08em
\lightrulewidth=.05em
\cmidrulewidth=.03em
\belowrulesep=.65ex
\belowbottomsep=0pt
\aboverulesep=.4ex
\abovetopsep=0pt
\cmidrulesep=\doublerulesep
\cmidrulekern=.5em
\defaultaddspace=.5em
```

Und einige interne Zähler, die für den Endnutzer von keinem Interesse sind, die unten beschrieben werden, sobald es nötig sein sollte:

```
\newcount\@cmidla
\newcount\@cmidlb
\newdimen\@aboverulesep
\newdimen\@belowrulesep
\newcount\@thisruleclass
\newcount\@lastruleclass
\@lastruleclass=0
\newdimen\@thisrulewidth
%  \end{macrocode}

% \DescribeMacro\futurenonspacelet
%
% Als n"achstes wird ein n"utzliches Makro definiert
% (mehr oder weniger direkt aus dem
% Kapitel Dirty Tricks aus dem \TeX book; dort verzeichnet). Man sollte
% =\futurenonspacelet= anstelle von =\futurelet= verwenden, wenn man nach dem
```

```
% n"achsten (non-space) Zeichen, nach einem Makro mit einem Argument sucht.
% (Nach einem Makro
% ohne ein Argument wird ein Leerzeichen ohnehin ignoriert, also w"urde
% =\futurenonspacelet= nicht gebraucht werden.) Dieser Kniff erlaubt es dem
% Nutzer, wei"se Leerzeichen zwischen aufeinanderfolgenden Linienbefehlen zu
% setzen (was in Version 1.00 nicht klappte).
% \begin{macrocode}
\def\futurenonspacelet#1{\def\@BTcs{#1}%
   \afterassignment\@BTfnslone\let\nexttoken= }
\def\@BTfnslone{\expandafter\futurelet\@BTcs\@BTfnsltwo}
\def\@BTfnsltwo{\expandafter\ifx\@BTcs\@sptoken\let\next=\@BTfnslthree
   \else\let\next=\nexttoken\fi \next}
\def\@BTfnslthree{\afterassignment\@BTfnslone\let\next= }
```

9.1 Linien von ganzer Stärke

Wenn man nicht in einer `longtable`Umgebung ist, kann man einfach Linien von ganzer Stärke als `\hrule` in einer `\noalign{}`-Gruppe bearbeiten. Innerhalb einer `longtable` jedoch muss die Linie wie eine `\cmidrule{1-\LT@cols}` gezogen werden (Die Begründung dafür ist in der `longtable`-Dokumentation erklärt).

Um beides zu erlauben, müssen alle Linienmakros sofort eine `\noalign`-Gruppe öffnen, während sie ausarbeiten, ob sie innerhalb einer `longtable` aufgerufen wurden; wenn das nicht gemacht wird, bekommt, der TEXzu Grunde liegender `\halign`-Prozess Schluckauf. Ich verwende den kleinen Trick von LATEX (`\ifnum=0‘}`), um dem Parser vorzumachen, dass die Klammerzahl in Ordnung ist. Die Klammer wird nach dem ganzen Überspringen am Ende des `\@BTendrule`-Makros dann tatsächlich geschlossen.

Die Klasse-1-Linien und `\specialrule` unterscheiden sich in den Voreinstellungen nur im Abstand darüber und darunter und in der Stärke, die durch eine normale Routine erlassen wird, `\@BTrule`, siehe unten. Die Abstände, `\@aboverulesep` und `\@belowrulesep`, werden innerhalb der `\noalign`gruppe festgelegt, werden also von `\@BTrule` übernommen. Genauso weiß `\@BTrule` alles, was es über die Routine, die es aufgerufen hat, wissen muss, durch das Untersuchen der übernommenen `\@thisruleclass`. Das optionale Stärkenargument wird von `\@BTrule` analysiert, nachdem es auf die Voreinstellung gesetzt wurde, falls es fehlt.

`\toprule`
`\midrule`
`\bottomrule`
`\specialrule`

```
\def\toprule{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=\abovetopsep
  \global\@belowrulesep=\belowrulesep
  \global\@thisruleclass=\@ne
  \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}
\def\midrule{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=\aboverulesep
  \global\@belowrulesep=\belowrulesep
```

```
    \global\@thisruleclass=\@ne
    \@ifnextchar[{\@BTrule}{\@BTrule[\lightrulewidth]}}
  \def\bottomrule{\noalign{\ifnum0=‘}\fi
    \@aboverulesep=\aboverulesep
    \global\@belowrulesep=\belowbottomsep
    \global\@thisruleclass=\@ne
    \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}
  \def\specialrule#1#2#3{\noalign{\ifnum0=‘}\fi
    \@aboverulesep=#2\global\@belowrulesep=#3\global\@thisruleclass=\tw@
    \@BTrule[#1]}
```

`\addlinespace` Ein `\addlinespace` ist im Wesentlichen eine Linie ohne Breite, keinem Abstand darüber und angegebem (oder voreingestelltem) Abstand darunter. Aber weil die Linie nicht wirklich gezogen wird, sondern bloßein `\vskip` ist, ist es nicht nötig, zu schauen, ob man in einer longtable ist. Also muss man `\@BTrule` nicht als ‘echte’ Linie bezeichnen. Aber wir teilen den `\@BTendrule` Vorausschau- und flagsetting -Code (sieh unten), und der `\vskip` wir dort gemacht.

```
\def\addlinespace{\noalign{\ifnum0=‘}\fi
  \@ifnextchar[{\@addspace}{\@addspace[\defaultaddspace]}}
\def\@addspace[#1]{\global\@belowrulesep=#1\global\@thisruleclass=\tw@
  \futurelet\@tempa\@BTendrule}
```

`\@BTrule` Alle Linien (außer `\addlinespace`) teilen diesen Code.

```
\def\@BTrule[#1]{%
  \global\@thisrulewidth=#1\relax
```

Das Stärkenargument (wenn der Nutzer keines angibt, dann wird die Aufrufroutine `\@BTrule` mit der Voreinstellung aufgerufen haben) sollte für spätere Verwendung in einer globalen Variable gespeichert werden.

```
  \ifnum\@thisruleclass=\tw@\vskip\@aboverulesep\else
```

Speziallinien fügen immer angegebenen Abstand darüber ein. (Es ist zu beachten, dass `addlinespaces` hier nicht gehen).

```
  \ifnum\@lastruleclass=\z@\vskip\@aboverulesep\else
  \ifnum\@lastruleclass=\@ne\vskip\doublerulesep\fi\fi\fi
```

Nach Text (der letzten Klasse-1-Linie) sollte der Linie `\aboverulesep` vorangehen; doch falls es direkt nach einer vorhergehenden Linie kommt, sollte ein `\doublerulesep` eingefügt werden.

Nun wird durch einen sehr fiesen Hack herausbekommen, ob man sich in einer `longtable` befindet. Das ist leicht, wenn `\longtable` nicht mal definiert ist: Dann ist man nicht drin. Aber es reicht nicht einfach nur zu gucken, ob `longtable` geladen ist – man könnte ja auch innerhalb einer normalen Tabelle anstatt einer `longtable` sein. Also wird geguckt, ob `\hline` von seiner LaTeXdefinition umgemodelt wurde, um genauso wie `\LT@hline` zu sein. (Momentan macht `longtable` diese Umdefinition, wenn es eine `longtable`-Umgebung öffnet, aber nicht global,

also ist es gelöscht, sobald sich die Umgebung schließt.) Ein weiteres Paket könnte das möglicherweise tun! Und `longtable` könnte sich verändern, insofern es das einführt! Bisher ist es nicht völlig sicher, aber ich habe noch keine bessere Methode gefunden.

Wir richten `\@BTswitch` ein, um `\@BTnormal` oder `\@BLTrule` aufzurufen, je nachdem was passt, und rufen es dann auf.

```
\ifx\longtable\undefined
   \let\@BTswitch\@BTnormal
\else\ifx\hline\LT@hline
   \let\@BTswitch\@BLTrule
\else
   \let\@BTswitch\@BTnormal
\fi\fi
\@BTswitch}
```

`\CT@arc@` Das ist Unterstützung für das `colortbl`-Paket für farbige Linien. `\CT@arc@` hält das `\arrayrulecolor` Setting.

```
\AtBeginDocument{%
   \providecommand*\CT@arc@{}}%% colortbl support
```

`\@BTnormal` Das ist für den Fall, dass wir *nicht* in einer `longtable` sind. Wir sind bereits in einer `\noalign`-Gruppe, alles, was getan werden muss, ist eine `\hrule` zu ziehen und alle zurückliegenden Abstände zu verschlingen, um dann die Abschlussroutine mit `\@tempa` mit dem nächsten Zeichen im Dokument gleichzusetzen.

```
\def\@BTnormal{%
    {\CT@arc@\hrule\@height\@thisrulewidth}%
    \futurenonspacelet\@tempa\@BTendrule}
```

`\@BLTrule` Das ist für eine Linie voller Stärke in einer `longtable`. Zuerst wird geguckt, ob ein Kerningargument verwendet wurde; falls das so ist, soll `\@@BLTrule` es auslesen, ansonsten kann `\@@BLTrule` mit einem leeren String aufgerufen werden:

```
\def\@BLTrule{\@ifnextchar({\@@BLTrule}{\@@BLTrule()}}
```

`\@@BLTrule`

```
\def\@@BLTrule(#1){\@setrulekerning{#1}%
\global\@cmidlb\LT@cols
```

Die Routine `\@setrulekerning` analysiert die Zeichen des Kerningarguments und setzt die globalen Kerningstärken entsprechend (oder zur Voreinstellung, wenn der Nutzer sie nicht explizit angegeben hat). Die globale Anweisung an `\@cmidlb` legt die Spaltenzahl für die `\@cmidrule`makro fest, die mit `cmidrules` geteilt wird.

```
\ifnum0=‘{\fi}%
```

Schließt die momentan geöffnete \noalign-Gruppe. Innerhalb einer longtable werden Linien als Bezugsstriche in einer Textbox, die \LT@cols Spalten breit ist, gezogen.

```
\@cmidruleb
```

Zeichnet die Linie. Der \@cmidruleb-Code wird mit gewöhnlichen \cmidrules geteilt.

```
\noalign{\ifnum0=‘}\fi
```

Ein neues noalign muss aber sofort geöffnet werden, da TEX sonst eine neue Textbox beginnen wird, wo wir keine wollen. Dann, nach dem Löschen des nicht gewollten weißen Platzes, kann die Abschlussroutine aufgerufen werden.

```
\futurenonspacelet\@tempa\@BTendrule}
```

\@BTendrule Wir schauen einen Schritt weiter; das Zeichen ist in \@tempa), um zu sehen, ob eine andere Linie folgt (Der Nutzer sollte sich schämen!). Wenn dem so ist, wird \@lastruleclass \@thisruleclass gleichgesetzt (und damit für die folgende Linie festgelegt). Wenn es keine darauffolgende Linie gibt, wird \@lastruleclass freigelassen (d. h. auf Null gesetzt), was technisch gesehen, nicht wahr ist, da gerade eine Linie gezogen wurde. Aber es stellt es korrekt für die nächste Linie, der begegnet wird, ein; die muss auf dazwischenliegenden Text folgen.

```
\def\@BTendrule{\ifx\@tempa\toprule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\midrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\bottomrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\cmidrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\specialrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\addlinespace\global\@lastruleclass=\@thisruleclass
  \else\global\@lastruleclass=\z@\fi\fi\fi\fi\fi\fi
  \ifnum\@lastruleclass=\@ne\relax\else\vskip\@belowrulesep\fi
  \ifnum0=‘{\fi}}
```

9.2 Spezielle Unterlinien

\@setrulekerning Der folgende Code analysiert die Trimmingargumente (wenn es welche gibt) für \cmidrule oder eine \BLTrule. Die Linie wird links und rechts durch \cmrkern@l und \cmrkern@l beschnitten, die beide als Voreinstellung Null haben, auf \cmidrulekern durch die einfachen Argumente (lr) gesetzt werden oder durch den Nutzer wie bei (r{.5em}) eingestellt werden. Es wird Zeichen für Zeichen durch die Argumente gestöbert. Die Zeichen r und l bewirken, dass \cmrkern@r oder \cmrkern@l auf \cmidrulekern gesetzt werden. Es gibt keine Vorschau, um zu sehen, ob eine Stärke das nächste Zeichen ist. Diese Strategie ist für die einfachen Befehle effizient, wohingegen es für die bedingten Befehle ineffizient ist. Viel wichtiger ist jedoch, dass es viel einfacher zu programmieren ist. Die Zeichen r und l legen auch \cmrswitch fest, sodass, wenn sich das nächste

Zeichen als {⟨*wd*⟩} herausstellt, das Kerning auf der Seite, die gerade festgelegt ist, gemacht wird. Ich war zu faul, um eine Fehlermeldung zu zu programmieren, falls anderen Zeichen als `r`, `l` oder {⟨*wd*⟩} begegnet wird.

```
\def\@setrulekerning#1{%
   \global\let\cmrkern@l\z@
   \global\let\cmrkern@r\z@
   \@tfor\@tempa :=#1\do
  {\def\@tempb{r}%
   \ifx\@tempa\@tempb
      \global\let\cmrkern@r\cmidrulekern
      \def\cmrsideswitch{\cmrkern@r}%
   \else
      \def\@tempb{l}%
      \ifx\@tempa\@tempb
         \global\let\cmrkern@l\cmidrulekern
         \def\cmrsideswitch{\cmrkern@l}%
      \else
         \global\expandafter\let\cmrsideswitch\@tempa
      \fi
   \fi}}
```

\cmidrule
\@cmidrule
\@@cmidrule
\@@@cmidrule

`\cmidrule` nutzt `\@lastruleclass` auf eine völlig andere Weise als die Linien von ganzer Stärke. (Vielleicht hätte ich ein anderes Flag nutzen sollen; es erschien mir zu der Zeit effizient ...). Das wird (links) auf eins gesetzt, wenn man sich in der Mitte einer Reihe von `\cmidrules` befindet oder eine neue (mit `\morecmidrules`) anfängt. Ansonsten, für den Fall, dass `\@lastruleclass` null ist, wird vor die Linie ein `\aboverulesep` gesetzt.

```
\def\cmidrule{\noalign{\ifnum0=‘}\fi
    \@ifnextchar[{\@cmidrule}{\@cmidrule[\cmidrulewidth]}}
\def\@cmidrule[#1]{\@ifnextchar({\@@cmidrule[#1]}{\@@cmidrule[#1]()}}
\def\@@cmidrule[#1](#2)#3{\@@@cmidrule[#3]{#1}{#2}}
```

Das hier ist ein Herumtüfteln, um die Voreinstellungen für fehlende optionale Argumente einzustellen. Es wird außerdem in einer anderen Reihenfolge zu `\@@@cmidrule` übergegangen, nämlich `[a-b]{width required}{kerning commands}` (das ist die Reihenfolge, in der die Argumente wirklich abgehandelt werden):

```
\def\@@@cmidrule[#1-#2]#3#4{\global\@cmidla#1\relax
    \global\advance\@cmidla\m@ne
    \ifnum\@cmidla>0\global\let\@gtempa\@cmidrulea\else
    \global\let\@gtempa\@cmidruleb\fi
    \global\@cmidlb#2\relax
    \global\advance\@cmidlb-\@cmidla
```

Das erstellt einen Schalter (`\@gtempa`), um die relvanten Routinen `\@cmidrulea` oder `\@cmidruleb` aufzurufen, abhängig davon, ob man von Spalte anfängt oder nicht.

```
    \global\@thisrulewidth=#3
```

Das wird durch Voreinstellung oder eingesetztes Argument eingestellt. Dann werden alle Trimmingargumente analysiert, um dementsprechend \cmrkern@r und \cmrkern@l einzustellen:

```
\@setrulekerning{#4}
```

Nun kann darüber, falls gewünscht, Platz einfügen, das \noalign geschlossen und dann zur angemessenen Linienziehroutine, wie oben beschrieben, gewechselt werden (\let bis \@gtempa):

```
\ifnum\@lastruleclass=\z@\vskip \aboverulesep\fi
\ifnum0=‘{\fi}\@gtempa
```

Nach gezogener Linie kann nun ein anderes \noalign geöffnet und die Abschlussroutine aufgerufen werden:

```
\noalign{\ifnum0=‘}\fi\futurenonspacelet\@tempa\@xcmidrule}
```

\@xcmidrule In dieser Abschlussroutine ist zu rpüfen, ob ein anderes \cmidrule folgt; wenn ja, sollte vertikal zurückgeschritten werden, damit es sich mit der gerade gezeichneten Linie fügt. Das Setzen von \@lastruleclass auf 1 wird verhindern, dass über dem nächsten wieder ein Abstand eingefügt wird. Wenn eine \morecimdrules folgt, wird ein (positives) \cmidrulesep eingefügt (und wieder einmal \@lastruleclass auf 1 gesetzt). Ansonsten ist das die letzte Linie der momentanen Gruppe und es lässt sich ganz einfach \belowrulesep dahintersetzen. Abschließend wird das \noalign geschlossen.

```
\def\@xcmidrule{%
   \ifx\@tempa\cmidrule
       \vskip-\@thisrulewidth
       \global\@lastruleclass=\@ne
   \else \ifx\@tempa\morecmidrules
       \vskip \cmidrulesep
       \global\@lastruleclass=\@ne\else
       \vskip \belowrulesep
       \global\@lastruleclass=\z@
   \fi\fi
   \ifnum0=‘{\fi}}
```

\@cmidrulea Dieser Code (wird unter diesem Text aufgerufen) zeichnet die Linien. Sie werden als Boxen im Text gezeichnet, anstatt in einer \noaligngruppe, was das Kerning links und rechts gestattet.

```
\def\@cmidrulea{%
   \multispan\@cmidla&\multispan\@cmidlb
   \unskip\hskip\cmrkern@l%
  {\CT@arc@\leaders\hrule \@height\@thisrulewidth\hfill}%
   \hskip\cmrkern@r\cr}%
```

\@cmidruleb

```
\def\@cmidruleb{%
    \multispan\@cmidlb
    \unskip\hskip \cmrkern@l%
   {\CT@arc@\leaders\hrule \@height\@thisrulewidth\hfill}%
    \hskip\cmrkern@r\cr}%
```

\morecmidrules Das ist wirklich ein hohler Befehl; die ganze Arbeit ist davor bereits in der \cmidruleroutine gemacht worden. Es lässt sich einen Schritt voraus schauen, um zu gucken, ob ein \morecmidrules der aktuellen \cmidrule folgt, und können das Flag setzen, falls ja. Ansonsten macht \morecmidrules an sich überhaupt nichts.

```
\def\morecmidrules{\noalign{\relax}}
```

⟨/package⟩

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.