

Dokumentation – Version 2.0.0

Autor & Instandhaltung:
Florian Sihler (florian.sihler@web.de)
20. September 2019

Abstract

Oder auch Einleitung für **VER 2.0.0**

Die L^AT_EX-Dokumentklasse **Lilly** ist im Rahmen des Studiums von Florian Sihler entstanden, und dient der Generierung studiumsrelevanter Dokumente & Mitschriften, in dessen Rahmen Lilly weiter angepasst und (hoffentlich) optimiert wurde. Die klassische Version basiert auf der **KOMA-Script** Dokumentklasse **scrbook**.

Das Ziel ist es ein Latexdokument direkt in verschiedenen Versionen zu generieren! Die aktuelle Version „2.0.0 - Jake ist auch nur Java“ besitzt den Status Work in Progress!

Inhaltsverzeichnis

1 Einleitung	1
1.1 Installieren von Lilly	1
1.1.1 Linux	
1.1.2 Windows WAR Ausstehend	
1.1.3 MacOS WAR Ausstehend	
1.1.4 Keine Installation	
1.2 Erstellen eines Dokuments mit Lilly VER 2.0.0	3
1.2.1 Das Gerüst	
1.2.2 Die Böxli	
1.2.3 Hyperlinks	
1.3 Einbinden von weiteren Dokumenten	5
1.3.1 Aufgliedern eines Dokuments	
1.3.2 Übungsblätter	
2 Mathe	8
2.1 Weitere Befehle	8
2.1.1 Operatoren	
2.1.2 Symbole	
2.1.3 Kompatibilität	
2.1.4 Shortcuts	
2.2 Plots VER 1.0.8	13
2.2.1 graph-Environment	
2.3 3D-Plots	16
3 Grafiken	17
3.1 Grundlegende Symbole	17
3.1.1 Die Ampeln	
3.1.2 Emoticons WAR Ausstehend	
3.1.3 Utility WAR Ausstehend	
3.2 Diagramme & Graphen	18
3.2.1 Graphen	
3.2.2 Rotation	
3.2.3 Automaten WAR Work in Progress	
3.2.4 Schaltkreise WAR Ausstehend	
3.2.5 Neuronen WAR Work in Progress	
3.2.6 ER-Diagramme	
3.2.7 Verzeichnis-Bäume WAR Work in Progress	
3.3 Mitgelieferte Grafiken	25
3.4 Zusätzliche Optionen	27
3.4.1 Externalisierung	

3.4.2 Platzhalter	
3.5 Weiterführende Symbole	27
3.5.1 Embleme	
4 Farben	30
4.1 Die normalen Farbprofile	30
4.1.1 Das Standardfarbprofil	
4.1.2 Das Druckprofil	
4.2 Farberweiterungen	33
4.3 Weitere Planungen	34
5 Listings	35
5.1 Die grundlegenden Eigenschaften	35
5.1.1 Grundlegendes Design	
5.1.2 Das MAIN-Paket	
5.1.3 Das MIPS-Paket	
5.1.4 Kontrolle der Sprachen	
5.2 Marker und weitere Befehle	45
5.2.1 Literates	
5.2.2 Marker	
5.3 Advanced Listings	46
5.4 Runtimes	46
6 Boxen	47
6.1 Grundlegendes	47
6.1.1 Eine kleine Einführung	
6.1.2 Der Box-Controller	
6.1.3 Die Boxmodi	
6.1.4 Standart-Boxen	
6.2 Info-Boxes	60
6.2.1 Wie es funktioniert	
7 Allzweckmodule und Kern	64
7.1 Die Allzweckmodule	64
7.1.1 Lilly-Befehle	
7.1.2 Kodierung	
7.1.3 Listen	
7.1.4 Zufall	
7.1.5 Kurzbefehle	
7.1.6 Fallunterscheidungen	
7.2 Der Kern	72
7.2.1 Booleans und Debug	
7.2.2 Vanilla	
7.2.3 Paket-Kontrolle	
7.2.4 Pfadverwaltung	
7.2.5 Inhaltskontrolle	
7.3 Der Keyval-Parser	76
8 Präsentatoren	79

8.1 Formatierungen	79
8.2 Ornamente	80
8.3 Tabellen	82
8.3.1 Iterationen	
8.3.2 Weitere Designs	
8.4 Gedichte	88
8.5 Stundenpläne	91
8.5.1 Komfort	
8.5.2 Universitäts Stundenpläne	
9 Controller	99
9.1 Umgebungen	99
9.2 Worttrennung	100
9.3 Verlinkungen	101
9.4 Modi-Kontrolle	102
9.5 Layout Kontrolle	102
9.5.1 Das Mitschrieb-Layout	
9.5.2 Das Übungsblatt-Layout	
9.5.3 Das Zusammenfassungs-Layout	
9.5.4 Das Plain-Layout	
9.5.5 Das ElegantBook-Layout	
9.5.6 Das Paper-Layout	
9.5.7 Das PnPGuide-Layout	
9.5.8 Das Poems-Layout	
9.6 Titelseiten	117
9.6.1 Zufällige Texte	
9.6.2 Philosopher	
10 Jake	123
10.1 Grundlegendes	123
10.1.1 Entwicklung	
10.1.2 Die Installation	
10.1.3 Lilly mit Jake installieren	
10.1.4 Jake im Überblick	
10.1.5 Entwicklerinformationen	
10.2 Gepard	128
10.2.1 Konfigurationsdateien	
10.2.2 Gepard Module im Allgemeinen	
10.2.3 Buildrules	
10.2.4 Expandables	
10.2.5 Hooks	
10.2.6 Name Maps	
11 Exkurse	135
11.1 Wie man sich eine eigene Vorlesung bastelt	135
12 Aussicht	139
12.1 Todos	139
12.2 Geplant für VER 2.1.0	139

13 Anhang	141
13.1 Version VER 1.0.7	141
13.1.1 Installation in Linux	
13.1.2 Spezifikation: Plots	
13.2 Version VER 1.0.9	143
13.2.1 Installation in Linux	
13.2.2 Installation in MacOS	



EINLEITUNG

INTEGRIEREN VON LILLY – DIE GRUNDLAGEN VON A-Z

1.1 Installieren von Lilly

Aktuell kommt die Dokumentklasse ohne .ins oder .dtx Datei, dafür allerdings mit einem Installer für alle Debian (Linux) basierten Betriebssysteme, an einer Variante für MacOS und Windows wird momentan gearbeitet.

VER 2.0.0

Bemerkung 1 – Mithilfe

Wenn du dich mit \TeX oder \LaTeX auskennst, schreibe an folgende Email-Adresse florian.sihler@web.de.

Mittlerweile gibt es auch ein offizielles Github-Repository (<https://github.com/EagleoutIce/LILLY>) über das die gesamte Entwicklung abläuft. Hier werden noch Helfer für folgende Aufgaben gesucht:

- ◊ Java - Entwicklung
- ◊ Bash, Konsolen - Entwicklung
- ◊ Kommentieren in Markdown
- ◊ Maintaining (\TeX , \LaTeX)
- ◊ Kommentieren in Doxygen
- ◊ Layout Gestaltung
- ◊ \TeX , \LaTeX -Entwicklung
- ◊ Tester (Ubuntu, Mac OS X, Windows)

1.1.1 Linux

Für Versionen < 2.0.0 klicke hier: [klick mich!](#)

Mit der Portierung von *Jake* in die Programmiersprache Java hat sich die Installation von LILLY, immens vereinfacht. Da man hierfür allerdings *Jake* benötigt, der sich dann um alles weitere kümmert, sei hier einmal nur kurz erklärt, wie man die `stable`-Version von Jake installiert, für mehr Infos siehe: [Jake Installieren](#).

Mit dem Bezug dieser Dokumentation sollte eine `jake.jar` Datei einhergegangen sein, die es nun gilt auszuführen. Natürlich wird hierfür Java benötigt, auf einem apt-Basierten Betriebssystem installiert man Java wie folgt:

```
sudo apt install default-jdk
```

Für alle anderen Derivate gilt es sich auf <https://www.oracle.com/de/java/> entsprechend zu informieren. Einmal installiert, genügt ein Ausführen der `.jar` Datei mithilfe von `java -jar jake.jar` oder durch einen Doppelklick, sofern die entsprechende `.jar` als ausführbar markiert ist. Bei der Installation gilt es die Angabe einer [Nutzerkonfiguration](#) zu beachten. zieht man bunte Fenster der Kommandozeile vor, so ist man in der Lage mit

`java -jar jake.jar GUI` eine grafische Unterstützung zu erhalten, die allerdings momentan noch in Arbeit und noch lange davon entfernt ist, dieselbe Mächtigkeit wie die Kommandozeile zu erreichen. Einmal installiert lässt sich *Jake* einfach durch `jake` verwenden.

Lilly mit Jake installieren

Nun genügt ein Ausführen von `jake install`, wobei mithilfe der Option `-lilly-path` der Pfad angegeben werden kann, an dem sich die LILLY.cls befindet:

```
jake install -lilly-path: '/absoluter/Pfad/zum/Lilly/Ordner'
```

Anschließend sollte es möglich sein Dokumente mit LILLY zu kompilieren. Gemeinsam mit LILLY werden eine Vielzahl an Beispieldokumenten ausgeliefert, die die Verwendung anschaulich machen sollen und somit auch als Test für eine erfolgreiche Installation verwendet werden können. Exemplarisch sei `test & bonus/map_tests/test.conf` genannt, welches auch die `\getGraphics`-Schnittstelle etabliert.

1.1.2 Windows WAR Ausstehend

1.1.3 MacOS WAR Ausstehend

1.1.4 Keine Installation

Bemerkung 2

Von dieser Methode wird abgeraten

Natürlich lässt sich Lilly auch so nutzen, hierfür muss einfach nur die zu kompilierende Latex-Datei im selben Verzeichnis wie die Datei `Lilly.cls` liegen (also: `Lilly`). Natürlich kann dies bei mehreren Dateien, die auf Lilly zugreifen, unübersichtlich werden.

1.2 Erstellen eines Dokuments mit Lilly VER 2.0.0

1.2.1 Das Gerüst

Es ist recht einfach ein Dokument mit Lilly zu erstellen. Da es sich ja um eine Dokumentklasse handelt, wird sie wie folgt eingebunden:

```
\documentclass[Dokumentation]{Lilly}
```

Für den Typ gibt es unter anderem 4 Optionen:

VER 1.0.7

- ◊ Dokumentation
- ◊ Uebungsblatt
- ◊ Mitschrieb
- ◊ Zusammenfassung

Mit VER 2.0.0 ist es nötig nur `Dokumentation` anstelle von `Typ=Dokumentation` zu schreiben, da die explizite Zuweisung versucht, auf die entsprechende Datei zu referenzieren. Die Definition für dieses Dokument lautet zum Beispiel:

```
\documentclass[Dokumentation]{Lilly}
```

In Kombination mit `Jake` ist es zudem noch möglich die Option `Jake` anzugeben, die es Jake gestattet die Dokumentspezifischen Parameter zu bestimmen.

Zu beachten ist, dass die anderen Optionen weitere Parameter *fordern*.

So benötigt `Mitschrieb` noch den Parameter `Vorlesung`, der zusammen mit dem Parameter `Semester` gemäß:

```
\input{\LILLYxPATHxDATA/Semester/\LILLYxSemester/Definitions/
      \LILLYxVorlesung}
```

die für die jeweilige Vorlesung definierten Daten lädt. Erklärungen für die geladenen Daten befinden sich in den jeweiligen README-Dateien:

1. Semester [..../Lilly/source/Data/Semester/1/Readme.md](#)

2. Semester [..../Lilly/source/Data/Semester/2/Readme.md](#)

Weiter nutzt `Uebungsblatt` ebenfalls `Vorlesung&Semester` sowie noch die optionale Option (tihih) `n` die angibt, um das wievielte Übungsblatt es sich handelt. Darüber müssen wir uns aber in der Regel keine Gedanken machen. Trägt unser Übungsblatt einen Namen wie `uebungablaatt-gdbs-42.tex`, so kann `Jake` über sogenannte NameMaps entsprechend alles konfigurieren, in diesem Fall benötigt dein Übungsblatt auch kein `documentclass` mehr, es genügt das direkte Schreiben von Latex-Code, der Rest wird von Jake übernommen.

Entsprechend des Dokumenttyps werden gegebenenfalls auch bereits etliche Seiten generiert, dies gilt es zu beachten, wenn man vielleicht nur etwas testen möchte. In diesem Fall gibt es (wie später auch noch weiter aufgeführt) den sogenannten *Bonustyp* `PLAIN`, welcher ein leeres Dokument erstellt!

1.2.2 Die Böxi

Jede Box besteht als Environment und lässt sich wie folgt nutzen:

Definition 1.1 – Titel

Hallo Welt

```
\begin{definition}[Titel]
    Hallo welt
\end{definition}
```

Satz 1.1 – Titel

Hallo Welt

```
\begin{satz}[Titel]
    Hallo welt
\end{satz}
```

Lemma 1.1

Hallo Welt

```
\begin{lemma}
    Hallo welt
\end{lemma}
```

Aufgabe 0.1 – Titel

3 Punkte

Hallo Welt

```
\begin{aufgabe}{Titel}{3}
    Hallo welt
\end{aufgabe}
```

Letztere ändert sich zum Beispiel mit dem Dokumenttyp, so wird die Aufgabenbox in einem Übungsblatt immer noch wie folgt veranschaulicht:

Aufgabe 2 – Titel

Hallo Welt

```
\begin{aufgabe}{Titel}{3}
    Hallo welt
\end{aufgabe}
```

Hier eine Liste aller Boxen:

- ◊ definition
- ◊ satz
- ◊ zusammenfassung
- ◊ bemerkung
- ◊ beweis
- ◊ aufgabe
- ◊ beispiel
- ◊ lemma
- ◊ uebungsblatt

Sie können alle mithilfe von:

```
1 %% Allgemein
2 % \def\LILLYxBOXx<FirstLetterUp–Name>xEnable{FALSE}
3 \def\LILLYxBOXxDefinitionxEnable{FALSE}
```

jeweils deaktiviert und damit aus dem Dokument entfernt werden (auch nur abschnittsweise, das Reaktivieren funktioniert analog mit TRUE).

Eine Auflistung ihrer lässt sich mit dem `\listof` Befehl erzeugen (*Die Bezeichnung der Listen sind bisher noch inkonsistent :/*). Beispielhaft:

```
\listofDEFINITIONS
```

erzeugt hierbei (*Natürlich sind die Linien nur zur Trennung eingefügt.*):

Alle Definitionen

1.1	Titel	4
6.1	Titel	49
6.2	¶ Titel	49

1.2.3 Hyperlinks

Eine Sprungmarke innerhalb eines Dokuments lässt sich mit:

VER 1.0.0

```
\elable{mrk:Hey} %% \elable{<Sprungmarke>}
```

erstellen. Referenziert werden kann sie mithilfe des Befehls `\jmark`:

```
\jmark[Klick mich]{mrk:Hey} %% \jmark[Text]{Sprungmarke}
```

der erzeugte Link: **Klick mich**, passt sich zudem der Akzentfarbe der aktuellen Boxumgebung und dem Druckmodus an:

Zusammenfassung 1.1 – Testzusammenfassung



Siehe hier: **Klick mich** (Wenn Druck: Klick mich^{→ 5})

Der alternative Vertreter für `\jmark` ist `\hmark`, er ignoriert sämtliche Farbattribute:

```
\hmark[Klick mich]{mrk:Hey} %% \hmark[Text]{Sprungmarke}
```

und erzeugt damit: **Klick mich**.

1.3 Einbinden von weiteren Dokumenten

1.3.1 Aufgliedern eines Dokuments

Um Dokumente portabel kompilierbar zu machen, setzt das Makefile gemäß der Konfiguration `\LILLYxPATH` (hier: „./“). Nun lässt sich mithilfe des Befehls `\linput{<Pfad>}` eine Datei relativ zur Quelldatei angeben (beachte, dass absolute Pfade bei `\linput` keinen Sinn machen. Hierfür solltest du weiterhin `\input` verwenden).

VER 1.0.4

Zudem lässt sich damit über `\LILLYxDOCUMENTxSUBNAME` der Name der zuletzt eingebundenen Datei (`Data/Einleitung.doc`) abfragen.

Weiter gilt zu beachten, dass es *nicht* möglich ist, das klassische `\include` zu verwenden! Dieser Befehl wird aber von LILLY deswegen direkt entsprechend erneuert (hierzu wird das klassische Latex `\input` im Zusammenspiel mit `\clearpage` verwendet, nicht LILLYs `\linput`!). Es ist also im Endeffekt doch möglich Dokumente mit `\include` zu verwenden.

VER 1.0.7

1.3.2 Übungsblätter

Da es von Bedeutung ist Übungsblätter so zu erstellen, dass die Abgaben direkt in die Mitschrift eingebunden werden können, gibt es hierfür eine einfache Möglichkeit:

```

1 %% \inputUB{<Name>}{<Nummer>}{<Pfad - linput>}
2 \inputUB{Mengen}{1}{Aufgaben_Data/Uebungsblatt_1.tex}
3
4 %% Wird zu:
5 \clearpage
6 \begin{uebungsblatt}[Mengen][1]
7   \linput{Aufgaben_Data/Uebungsblatt_1.tex}
8 \end{uebungsblatt}
9 \newpage

```

Übungsblätter sind nur in **complete**-Varianten verfügbar, werden also sonst nicht eingebunden!

Ein Übungsblatt erstellen

Doch wie erstellt man nun ein fachgerechtes Übungsblatt? Nun, da es sich hier um die Schnelleinführung handelt, ein paar Vorgaben. Benenne dein Übungsblatt nach dem Schema:

`uebungsblatt-<VORLESUNG>-<BLATTNUMMER>.tex`

Die Reihenfolge spielt keine Rolle, ein beispielhafter Name könnte sein:
`gdb-uebungsblatt-13.tex`. Nun erstelle dir eine `jake.conf`-Datei, wobei egal ist wie sie heißt, solange sei auf `.conf` endet (fürs Autocomplete ☺). In sie trägst du folgendes ein:

```

1 file      = @[SELTEXF]
2 operation = file_compile
3
4 lilly-modes = uebungsblatt
5
6 lilly-show-boxname = false
7
8 lilly-nameprefix = FlorianS-Partner-
9 lilly-author = Florian Sihler, Mein Partner
10
11 lilly-n = @[AUTONUM]

```

Natürlich kannst du die Namen entsprechend ändern. Das sieht jetzt aus wie viel, aber das musst du nur einmal machen, sofern du die Konfigurationsdatei immer in das Verzeichnis mitkopierst, indem sich die Übungsblatt `.tex` und **nur** diese `.tex`-Datei befindet. Wir werden uns später mit besseren Konfigurationen beschäftigen, die keinerlei Nachaufwand benötigen und galanter sind. In das Übungsblatt können wir nun unsere Aufgaben stecken. Hier ist der *gesamte* Inhalt der oben genannten `TeX`-Datei:

```

1 \begin{aufgabe}{Tolle Aufgabe}{400} % 400 Punkte
2   Die Aufgabenbeschreibung, blah, blah, blah, \ldots
3   \begin{aufgaben}
4     \item Teilaufgabe a)
5     \item Teilaufgabe b)
6     \item \ldots
7   \end{aufgaben}
8 \vSplitter

```

```
9 \begin{aufgaben}
10   \item Antwort zu Teilaufgabe a)
11   \item Antwort zu Teilaufgabe b)
12   \dots
13 \end{aufgaben}
14 \end{aufgabe}
15
16 %% Weitere Aufgaben, wenn gewünscht
```

Kompilieren kann man den Spaß nun mit: `jake jake.conf`. Und das wars, Boom ☺:

Aufgabe 3 – Tolle Aufgabe

Die Aufgabenbeschreibung, blah, blah, blah, ...

a) Teilaufgabe a)

b) Teilaufgabe b)

c) ...

a) Antwort zu Teilaufgabe a)

b) Antwort zu Teilaufgabe b)

c) ...

MATHE

EINZELNE VARIATIONEN UND EINE MENGE ABKÜRZUNGEN

VER 2.0.0

2

An sich ändert LILLY nicht viel an der normalen Implementation der Mathewelt. Dieses Paket liegt hier:

`\LILLYxPATHxMATHS` = source/Maths

◊ `\LILLYxMathxMode`

v1.0.3

Der verwendete Mathemodus lässt sich mithilfe des Befehls `\LILLYxMathxMode` frei einstellen. Standardmäßig wird dieser Wert auf *normal* gesetzt.

Bemerkung 3 – Standalone-Math

Mit VER 2.0.0 wurde die Mathe-Integration als eigenes Paket LIB `LILLYxMATH` etabliert, welches sich eigenständig über

`\usepackage{LILLYxMATH}`

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

2.1 Weitere Befehle

2.1.1 Operatoren

Diese Definitionen befinden sich in der Datei: Maths/_LILLY_MATHS_OPERATORS. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB `LILLYxMATH` geladen.

◊ `\overbar{text}`

v1.0.3

Lilly liefert den Befehl auf Basis von `\mkern` so, dass er direkt Abstände zwischen den Overlines definiert, sodass kein manueller Abstand eingefügt werden muss. So ergibt sich:

<code>\overbar{a_1}</code>	<code>\overbar{a_2}</code>	$\overline{a_1} \overline{a_2}$
<code>\overline{a_1}</code>	<code>\overline{a_2}</code>	$\overline{a_1} \overline{a_2}$

◊ `\das, \sad, \daseq, \qesad, \shoudeq`

v1.0.3

Für Definitionen gibt es die Befehle `\das` (\doteq), `\sad` (\doteqdot), `\daseq` ($\doteq\!\doteq$), `\qesad` ($\doteq\!\doteq\!$) sowie `\shoudeq` ($\doteq\!\!\!$). All diese Befehle funktionieren sowohl in einer Matheumgebung, das auch im normalen text, sie werden mit `\ensuremath` abgesichert!

Bis auf den letzten werden zudem alle Befehle mithilfe von `\vcentcolon` realisiert.

◊ `\sqrt[n]{math-Ausdruck}`

v1.0.3

Weiter wurde das Aussehen der Wurzel verändert und die Möglichkeit hinzugefügt, über das optionale Argument „n“ höhere Wurzeln zu Formulieren, wir erhalten folgendes:

<code>\sqrt[3]{42}</code>	$\sqrt[3]{42}$
<code>\oldsqrt[3]{42}</code>	$3\sqrt{42}$

◊ `\det, \adj, \LH, \eig, \Dim, \sel, \sign, \diag, \LK, \rg, \KER, \Eig` v1.0.3

Diese vereinfachenden Operatoren solles es ermöglichen Schneller verschiedene mathematische Operatoren zu setzen

- | | | |
|------------------------------------|-----------------------------|------------------------------------|
| ◊ <code>\det</code> (det) | ◊ <code>\Dim</code> (dim) | ◊ <code>\LK</code> (<i>LK</i>) |
| ◊ <code>\adj</code> (adj) | ◊ <code>\sel</code> (SEL) | ◊ <code>\rg</code> (<i>rg</i>) |
| ◊ <code>\LH</code> (<i>LH</i>) | ◊ <code>\sign</code> (sign) | ◊ <code>\KER</code> (<i>ker</i>) |
| ◊ <code>\eig</code> (<i>Eig</i>) | ◊ <code>\diag</code> (diag) | ◊ <code>\Eig</code> (<i>Eig</i>) |

◊ `\Im, \mod, \Re, \emptyset` v1.0.2

Auch wurde das Aussehen von `\mod`, `\Im`, `\Re` und `\emptyset` modifiziert:

- | | |
|------------------------------|--|
| ◊ <code>\mod</code> (MOD) | ◊ <code>\Re</code> (\Re) |
| ◊ <code>\Im</code> (\Im) | ◊ <code>\emptyset</code> (\emptyset) |

◊ `\inf, \sup, \min, \max` v1.0.6

Auch hierbei handelt es sich wieder um stupide Abbildungen im Operator-Style:

- | | |
|---------------------------|---------------------------|
| ◊ <code>\inf</code> (inf) | ◊ <code>\min</code> (min) |
| ◊ <code>\sup</code> (sup) | ◊ <code>\max</code> (max) |

◊ `\abs{math-Ausdruck}` v1.0.9

Dieser Befehl vereinfacht das Schreiben von Betragsstrichen. Diese passen sich zudem automatisch an die vertikalen Dimensionen des Ausdrucks an:

<code>\$\abs{\frac{\pi-x^2}{\log 3x}}\$</code>	$\left \frac{\pi - x^2}{\log 3x} \right $
<code>\$ \frac{\pi-x^2}{\log 3x} \$</code>	$\left \frac{\pi - x^2}{\log 3x} \right $

◊ `env@matrix[Spaltendefinition], env@pmatrix[Spaltendefinition]` v1.0.2

Des Weiteren wurde noch die Matrixumgebung (`\env@matrix`) so erweitert, dass sie als optionales Argument eine gültige Array-Spaltendefinition entgegennimmt:

<pre> 1 \$\begin{pmatrix} cc c 2 1 & 2 & 3 \\ 3 4 & 5 & 6 4 \end{pmatrix}\$ </pre>	$\left(\begin{array}{cc c} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array} \right)$
--	---

◊ `\val`, `\sch`, `\dom`, `\grad`

v1.0.8

Auch hier handelt es sich um weitere Mathe-Operatoren, die selbstredend implementiert werden:

- ◊ `\val` (val)
- ◊ `\sch` (sch)

- ◊ `\dom` (dom)
- ◊ `\grad` (Grad)

◊ `\arccot`

v1.0.8

Da der ach so wichtige Arkuskotangens erstaunlicherweise nicht standardmäßig dabei ist, hier: `\arccot` (arccot).

◊ `\dif`, `\dint[Variable=<x>]`

v2.0.0

Auch hierbei handelt es sich wieder um stupide Abbildungen im Operator-Style für Integration und Differenzierung

- ◊ `\dif` (d)

- ◊ `\dint` ($\frac{d}{dx}$)

2.1.2 Symbole

Diese Definitionen befinden sich in der Datei: `Maths/_LILLY_MATHS_SYMBOLS`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxMATH` geladen.

◊ `\N`, `\Z`, `\Q`, `\R`, `\C`

v1.0.0

Für die einzelnen Zahlenräume werden einige Befehle zur Verfügung gestellt, die alle über `\ensuremath` abgesichert sind: `\N` (\mathbb{N}), `\Z` (\mathbb{Z}), `\Q` (\mathbb{Q}), `\R` (\mathbb{R}), `\C` (\mathbb{C}). Sie werden mithilfe von `\mathbb{B}` generiert.

◊ `\i`

v1.0.1

Die komplexe Einheit `i` wird mit `\i` zur Verfügung gestellt.

◊ `\epsilon`, `\phi`

v1.0.3

Weiter wurden die griechischen Buchstaben Epsilon und Phi modifiziert:

<code>\oldepsilon</code>	ϵ	<code>\epsilon</code>	ε
<code>\oldphi</code>	ϕ	<code>\phi</code>	φ

◊ `\B`, `\X`, `\K`, `\P`, `\F`, `\O`

v1.0.3

Zudem wird zum Beispiel die Menge der Binärzahlen über `\B` (\mathbb{B}), die chromatische Zahl über

\X (χ) und der generelle Körper mit \K (\mathbb{K}) zur Verfügung gestellt. Für die Potenzmenge liefert LILLY \P (\mathcal{P}), für die Menge der Funktionen \F (\mathcal{F}) und für die Groß-O-Notation \O (\mathcal{O}).

◊ `\join, \leftouterjoin, \rightouterjoin, \fullouterjoin`

v2.0.0

Da auch die Relationenalgebra Teil der Mathematik ist, hier die entsprechenden Symbole für die Joins:

◊ `\join` (\bowtie)

◊ `\leftouterjoin` ($\bowtie\!\!\!\leftarrow$)

◊ `\rightouterjoin` ($\bowtie\!\!\!\rightarrow$)

◊ `\fullouterjoin` ($\bowtie\!\!\!\leftrightarrow$)

Bemerkung 4 – Weitere Symbole

Weiter bindet LILLY das `pifont` Paket ein und liefert so zum Beispiel `\ding{51}` (✓) und `\ding{55}` (✗).

2.1.3 Kompatibilität

Diese Definitionen befinden sich in der Datei: `Maths/_LILLY_MATHS_COMPATIBILITIES`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von `\lib LILLYxMATH` geladen.

Hier werden einige Befehle eingerichtet, die entweder noch nicht zugeordnet wurden **VER 2.0.0** oder während der Vorlesung (im Überlebenskampf :P) ins `eagleStudiPackage` eingebaut worden sind.

◊ `\enum{items}, \liste{items}`

v1.0.0

Hier befinden sich die für *Lineare Algebra* kreierten: `\enum{items}` (`enumerate` mit `\narrowitems`) und `\liste{items}` (`enumerate` mit römischen Zahlen und `\narrowitems`).

◊ `\xa, \xb, \xc`

v1.0.1

Weiter existieren die Befehle `\xa` ($\overline{x_1}$), `\xb` ($\overline{x_2}$), `\xc` ($\overline{x_3}$), welche einen etwas größeren Abstand für eine bessere Lesbarkeit einfügen.

◊ `\crossAT{(PosX,PosY)}, \circAT{(PosX,PosY)}, \bblock{(PosX,PosY)}{text}`

v1.0.1

Für TikZ gibt es noch die Befehle `\crossAT{(PosX,PosY)}` (✗^a) und analog `\circAT{(PosX,PosY)}` (○^b), sowie `\bblock{(PosX,PosY)}{text}` (42^c). Hier fragt man sich nun vielleicht, warum diese nicht in einem entsprechenden TikZ-Paket sind. Im Rahmen der mit **VER 2.0.0** eingeführten Modularisierung hat sich diese Verteilung als günstig erwiesen.

◊ `env@nstabbing, env@centered, env@sqcases`

WAR Veraltet

v1.0.2

Weiter werden drei (mittlerweile obsolete) Umgebungen definiert:

^a`\tikz{\crossAT{(0,0)}};` – Zum Erhalt der Textzeile vertikal um $-0.35\baselineskip$ verschoben.

^b`\tikz{\circAT{(0,0)}};`

^c`\tikz{\bblock{(0,0)}{42}};` – Wieder vertikal um $-0.2\baselineskip$ verschoben.

- ◊ `env@nstabbing`: `tabbing`-Umgebung, ohne Abstände
- ◊ `env@centered`: `center`-Umgebung, ohne Abstände
- ◊ `env@sqcates`: Ähnelt `cases` - nur mit ']'.

- ◊ `\VRule{width}` v1.0.4

Zudem definiert sich noch für Tabellen der Befehl `\VRule{width}`, welcher eine Spalte variabler Größe für Tabellen zur Verfügung stellt. Eine exemplarische Verwendung findet sich hier:

```
1 \begin{tabular}{c!{\VRule[6pt]}c}
2   \specialrule{2pt}{0pt}{0pt}
3   You're my & Wonder Wall\\
4   \specialrule{2pt}{0pt}{0pt}
5 \end{tabular}
```

- ◊ `\trenner` WAR Veraltet v1.0.0

Fügt einen großen senkrechten Strich ein: `\trenner` (|).

2.1.4 Shortcuts

Diese Definitionen befinden sich in der Datei: `Maths/_LILLY_MATHS_SHORTCUTS`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxMATH` geladen.

Hier befinden sich einige abkürzende Befehle, die primär das Schreiben beschleunigen sollen. Sie werden auf Bedarf stetig erweitert.

- ◊ `\folge[Folgenglied=<a>]` v1.0.7

Setzt eine Folge, welche mit dem Index n arbeitet: `\folge` $((a_n)_{n \in \mathbb{N}})$.

- ◊ `\reihe[Folgenglied=<a_k>][Start=<0>]` v1.0.7

Setzt eine Reihe über die Glieder *Folgenglied* an *Start*: `\reihe` $(\sum_{k=0}^{\infty} a_k)$

- ◊ `\obda`, `\Obda` v1.0.8

Schreibt entsprechend o.B.d.A (`\obda`) und O.B.d.A. (`\Obda`) und beschleunigt damit das Tippen von Beweisen ☺.

- ◊ `\gdw`, `\limn`, `\sumn`, `\limk`, `\sumk` v1.0.7

Setzt verschiedene mathematische Ausdrücke:

◊ <code>\gdw</code> (\Leftrightarrow)	◊ <code>\sumn</code> ($\sum_{n=0}^{\infty}$)	◊ <code>\sumk</code> ($\sum_{k=0}^{\infty}$)
◊ <code>\limn</code> ($\lim_{n \rightarrow \infty}$)	◊ <code>\limk</code> ($\lim_{k \rightarrow \infty}$)	

- ◊ `\x[spacing=<~>]`, `\y[spacing=<~>]`, `\z[spacing=<~>]` WAR Veraltet v1.0.2

Setzt entsprechend: x , y und z .

◊ `\ceil[math-Ausdruck], \floor[math-Ausdruck]`

v2.0.0

Verkürzt das Schreiben von: `\left\lfloor \mathrm{Ausdruck} \right\rfloor` beziehungsweise `\lceil` & `\rceil` entsprechend:

◊ `\ceil(\lceil \frac{a}{b} \rceil)`

◊ `\floor(\lfloor \frac{a}{b} \rfloor)`

2.2 Plots VER 1.0.8

Für die Spezifikationen siehe hier: [klick mich!](#)

Diese Definitionen befinden sich in der Datei: Maths/_LILLY_MATHS_PLOTS. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxMATH` geladen.

◊ `\plotline[Farbe=<Ao>][Variable=<\mathbf{x}>]{Term}[offset=<\theta>]`

v1.0.8

Zeichnet in eine **Graph-Umgebung** eine Funktion (siehe Umgebung für Beispiel). Existiert auch außerhalb von `env@graph`, ist aber hier nur eingeschränkt nutzbar. Mit `offsetv2.0.0` lässt sich die Funktion entsprechend verschieben.

◊ `\plotseq[Farbe=<Ao>][Variable=<\mathbf{x}>]{Term}[Obergrenze=<maxX>][Untergrenze=<1>][Dicke=<1pt>]`

v1.0.8

Zeichnet in eine **Graph-Umgebung** eine Folge zwischen *Unter-* und *Obergrenze* mit Punkten der Größe *Dicke* (siehe Umgebung für Beispiel). Existiert auch außerhalb von `env@graph`, ist aber hier nur eingeschränkt nutzbar.

◊ `\xmark{text=<x>}{PosX}[linelength=<0.15>]`

v2.0.0

Setzt einen Marker auf der x -Achse bei $PosX$ mit dem text $text$. Für ein Beispiel, siehe **Graph-Umgebung**.

◊ `\ymark{text=<y>}{PosY}[linelength=<0.15>]`

v2.0.0

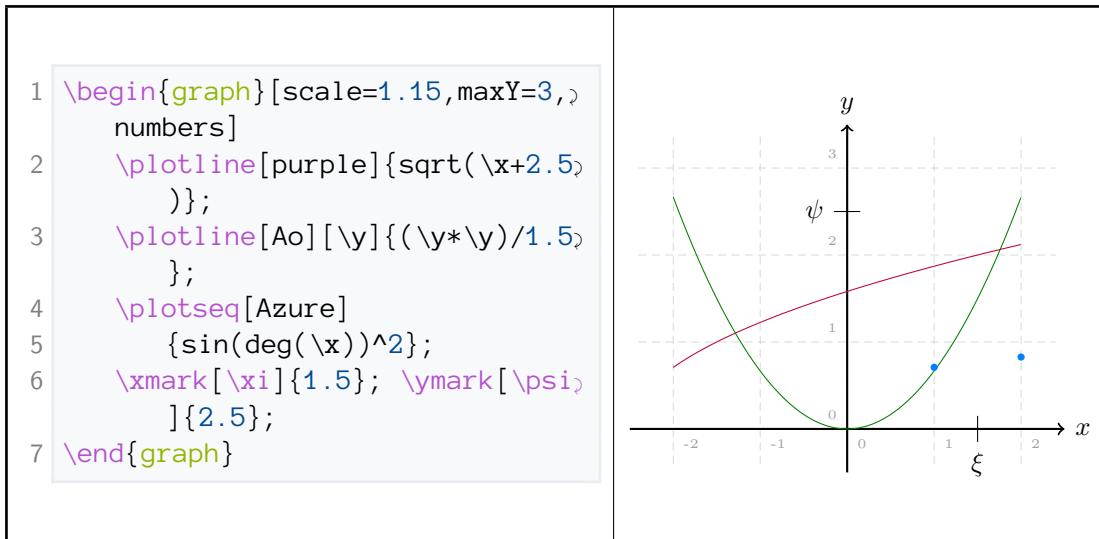
Setzt einen Marker auf der y -Achse bei $PosY$ mit dem text $text$. Für ein Beispiel, siehe **Graph-Umgebung**.

2.2.1 graph-Environment

◊ `env@graph[Konfigurationen][Tikz-Argumente]`

v1.0.8

Es existiert die folgende Implementation der Graph-Umgebung:



Für die *Konfiguration* gibt es die folgenden Parameter:

Bezeichner	Typ	Standard	Beschreibung
<code>scale</code>	<i>Zahl</i>	1	Skalierungsfaktor
<code>xscale</code>	<i>Zahl</i>	1	x -Skalierungsfaktor <small>v2.0.0</small>
<code>yscale</code>	<i>Zahl</i>	1	y -Skalierungsfaktor <small>v2.0.0</small>
<code>minX</code>	<i>Zahl</i>	-2	X-Achse Start
<code>maxX</code>	<i>Zahl</i>	2	X-Achse Ende
<code>minY</code>	<i>Zahl</i>	0	Y-Achse Start
<code>maxY</code>	<i>Zahl</i>	4	Y-Achse Ende
<code>offset</code>	<i>Zahl</i>	0.4	Zusatzlänge Achsen
<code>loffset</code>	<i>Zahl</i>	0.1	Unbeachteter Zusatz Achsen
<code>labelX</code>	<i>String</i>	<code>\$x\$</code>	Bezeichner X-Achse
<code>labelY</code>	<i>String</i>	<code>\$y\$</code>	Bezeichner Y-Achse
<code>samples</code>	<i>Zahl</i>	250	Anzahl an Kalkulationen
<code>numbers</code>	<code><></code>	false	Zeigt Zahlen an
<code>numXMin</code>	<i>Zahl</i>	0	Nummernstart x
<code>numYMin</code>	<i>Zahl</i>	0	Nummernstart y
<code>numbersize</code>	<i>Zahl</i>	5	Schriftgröße Nummerierung
<code>labelsize</code>	<i>Zahl</i>	10	Schriftgröße Texte

◊ **env@egraph**[Konfigurationen]v2.0.0

Funktioniert analog zu **env@egraph**, erlaubt allerdings keine weiteren *Tikz-Argumente*, sondern macht von **env@tikzternal** gebrauch, kann also ausgelagert werden. (TODO: LINK)

◊ **env@wgraph**{Ausrichtung}[Konfigurationen][Tikz-Argumente]v1.0.8

[*wrapfig-Zusatz*] [*width=⟨0pt⟩*]

Um die Graph-Umgebung noch vielfälder zu Gestalten wurde **env@wgraph** geschaffen. Nach reichlicher Überlegung wurde ein neuer Befehl etabliert anstelle es in das normale **graph**-Environment einzubetten. Er funktioniert mit der Syntax:

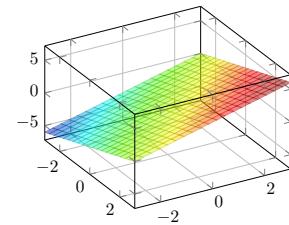
```
1 \begin{wgraph}{1}[][\caption{Wichtiger Graph}][400pt]
2   \plotline{\x*\x}
3 \end{wgraph}
```

2.3 3D-Plots

Bisher sind noch keine Definitionen für 3-Dimensionale Plots integriert. Deswegen hier die exemplarische Definition eines 3D-Plots:

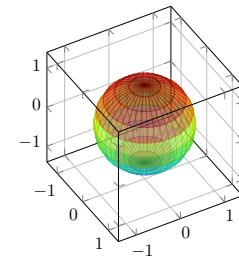
```

1 \begin{tikzexternal}[scale=0.5]
2 \begin{axis}[3d box=complete, colormap/bluered,
3     grid=major,view={60}{40},
4     z buffer=sort, data cs=polar]
5     \addplot3[data cs=cart,surf,domain=-3:3,,
6         samples=20, opacity=0.5] {x+y};
7 \end{axis}
8 \end{tikzexternal}
```



```

1 \begin{tikzexternal}[scale=0.6]
2 \begin{axis}[3d box=complete, axis equal ,
3     image, colormap/bluered,grid=major,view,
4     ={60}{40},z buffer=sort,enlargelimits=0,
5     .2,scale=2.3]
6     \addplot3[%
7         opacity = 0.5, surf,
8         samples = 21, variable = \u,
9         variable y = \v, domain = 0:180,
10        y domain = 0:360,
11    ]
12     ({cos(u)*sin(v)}, {sin(u)*sin(v)},
13     {cos(v)});
14 \end{axis}
15 \end{tikzexternal}
```



3

GRAFIKEN

ETLICHE VEREINFACHUNGEN UND ANDERE FREUDEN :D

VER 1.0.2

Dieses Paket liegt hier:

`\LILLYxPATHxGRAPHICS = source/Graphics`

Bemerkung 5 – Standalone-Graphics

Mit `VER 2.0.0` wurde die Grafik-Integration als eigenes Paket `LILLYxGRAPHICS` etabliert, welches sich eigenständig über

```
\usepackage{LILLYxGRAPHICS}
```

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

Dieses Paket basiert übrigens auf `LILLYxTIKZxCORE`, welche selbst nicht direkt sondern nur durch die hier verfügbaren Befehle dokumentiert ist und auch nicht frei verwendet werden sollte (ausgenommen natürlich, man weiß, wie ☺).

3.1 Grundlegende Symbole

Diese Definitionen befinden sich in der Datei: `source/Graphics/Tikz-Core/_LILLY_TIKZ_SYMBOLS`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxGRAPHICS` geladen.

Dieses Paket liefert grundlegende, mal mehr und mal weniger, nützliche Tikz-Grafiken, welche zum Großteil aus denen in der Vorlesung verwendeten Grafiken entstanden sind. Alle diese Grafiken benötigen TikZ (<https://www.ctan.org/pkg/pgf>).

◊ `\rectat{point}, \crectat{point}{color}`

v1.0.0

Ersterer Befehl setzt in einem `env@tikzpicture` ein (rot-)farbiges Rechteck, der zweite erlaubt die Auswahl der jeweiligen Farbe:

```
1 \begin{tikzpicture}
2   \rectat{(0,1)};
3   \crectat{(0,0)}{AppleGreen};
4 \end{tikzpicture}
```



3.1.1 Die Ampeln

Diese Definitionen befinden sich in der Datei: `.../Tikz-Core/_LILLY_TIKZ_AMPELN`. An sich handelt es sich hierbei um ein kleines Shortcut-Sammelsurium für Ampeln:

◊ `\ampelG`, `\ampelY`, `\ampelR`, `\ampelH` v1.0.2

Explizit verwendet werden diese Befehle in zum Beispiel in den Erklärungen zum Moore-&Mealy-Automaten auf Basis der Ampelschaltung (● ○ ○):

- ◊ `\ampelG` (●)
- ◊ `\ampelY` (○)
- ◊ `\ampelR` (●)
- ◊ `\ampelH` (○)

3.1.2 Emoticons [WAR Ausstehend]

Dieses Paket soll weitere lustige Begleiter im Textgeschehen zur Verfügung stellen:

- ◊ `\Ninja` (●)
- ◊ `\Smiley` (○)
- ◊ `\Sadey` (○)
- ◊ `\Xey` (○)
- ◊ `\Innocey` (○)
- ◊ `\Walley` (○)
- ◊ `\dSadey` (○)
- ◊ `\Fire` (○)
- ◊ `\Autumntree` (○)

3.1.3 Utility [WAR Ausstehend]

Dieses Paket soll die bisher von FontAwesome verwendeten Symbolen ersetzen und durch eigens erstellte Grafiken ersetzen.

3.2 Diagramme & Graphen

3.2.1 Graphen

Diese Definitionen befinden sich in der Datei: `source/Graphics/Tikz-Core/_LILLY_-TIKZ_GRAPHEN`. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LILLYxGRAPHICS geladen.

Bemerkung 6 – Motivation

Dieses Paket liefert grundlegende, mal mehr und mal weniger, nützliche Tikz-Grafiken, welche zum Großteil aus denen in der Vorlesung verwendeten Grafiken entstanden sind. Alle diese Grafiken benötigen TikZ (<https://www.ctan.org/pkg/pgf>).

◊ `\POLYRAD` {length} v1.0.2

Grundlegend wird für den Radius aller Polygone empfohlen `\POLYRAD` zu verwenden (Standardmäßig: 1.61cm).

Weiter definiert diese Bibliothek etliche sogenannte graphdots, welche alle nur in einer tikzpicture-Umgebung funktionieren, allen voran die Ur-Funktion:

◊ `\graphdot{fill-color}{(PosX,PosY)}{node-name}{border-color}`, v1.0.2
`\tgraphdot{fill-color}{(PosX,PosY)}{node-name}{border-color}`

Die Befehle unterscheiden sich darin, dass der `\tgraphdot` das Farbargument ignoriert und entsprechend transparent (`fill opacity = 0`) als Füllfarbe verwendet:

\graphdot{DebianRed}{(0,0)}{42}{a}{Azure}	(42)
\tgraphdot{DebianRed}{(0,0)}{42}{a}{Azure}	(42)

- ◊ \oragraphdot, \bluegraphdot, \greengraphdot, \purplegraphdot,
\goldgraphdot, \blackgraphdot, \norgraphdot, \margraphdot

v1.0.2

Alle weiteren graphdots sind nun nichts weiteres als Shortcuts für die eben genannten Befehle und besitzen die Signatur: \oragraphdot{(PosX, PosY)}{Text}{node-name}:

- ◊ \oragraphdot (42) ◊ \purplegraphdot (42) ◊ \norgraphdot (42)
- ◊ \bluegraphdot (42) ◊ \goldgraphdot (42) ◊ \margraphdot (42)
- ◊ \greengraphdot (42) ◊ \blackgraphdot (42)

Zur Information, alle diese Befehle wurden wie folgt präsentiert:

1 \tikz\graphdot{(0,0)}{42}{a};

wobei `\graphdot` entsprechend ersetzt wurde, weiter wurde für den Textfluss noch die Boxposition angepasst, dies spielt allerdings für den Graphen keine Rolle. Mit VER 2.0.0 wurden die Farben der Dots der neuen Palette entsprechend portiert.

- ◊ \graphPOI{(PosX, PosY)}{accent-color}{year}{obj-name}{brief}
{img-link}{extra}

v1.0.4

Präsentiert ein Timeline Point-of-interest, der schnell einen einheitlichen look für Timelines garantiert. Im Folgenden eine repräsentation, die den Wirrwarr an Optionen etwas übersichtlicher macht. Es gilt zu beachten, dass `\extra` hier die Rolle des entsprechendes Landes einnimmt:

<pre>\begin{tikzexternal}[scale=0.75, every node/.style={transform shape}] \graphPOI{(0,0)}{purple}{1999 n.Chr.} {Florian Sihler} {Florian Sihler ist der Autor dieses Dokuments.} [Data/2003.jpg] [https://github.com/EagleoutIce/Quickblit] [Deutschland]; \end{tikzexternal}</pre>	<p>● 1999 n.Chr. Deutschland Florian Sihler: Florian Sihler ist der Autor dieses Dokuments.</p> 
---	--

Hier wurde aus Platzgründen die Größe angepasst. Es gibt auch noch \LILLYxMODExEXTRA der es ermöglicht den \graphPOI-Befehl einzuschränken. Wir dieser Befehl auf \true (TRUE) gesetzt, so wird \graphPOI so konfiguriert, dass die zugehörige Grafik angezeigt wird. Ist dies nicht der Fall (in anderen Worten: \LILLYxMODExEXTRA=\false), so wird kein Bild angezeigt (auch der Link existiert dann nicht). Diese Version wurde erstellt um Urheberrechtsverletzungen zu vermeiden.

- ◊ \PgetX{Point}{out:x-cord}{out:y-cord},
\PgetX{out:x-cord}, \PgetY{out:y-cord}

v2.0.0

Da es oft notwendig ist die Koordinate eines Punktes weiter zu benutzen und da das Kreuzen von Koordinaten nervig ist, gibt es verschiedene Befehle die es erlauben, die entsprechenden

Koordinaten zu speichern, wobei die letzteren beiden nur lesbarere Alternativen für die erste sind, sofern die entsprechend andere Koordinate nicht benötigt wird:

```

1 \begin{tikzternal}
2   \node (A) at (1,2) {A};
3   \PgetXY{(A)}{\myX}{\myY};
4   % Befehle werden gebunden
5   \node (B) at (\myX,0) {B};
6   \PgetY{(B)}{\anotherY};
7   \node (C) at (1.5*\myY,\anotherY) {C};
8 \end{tikzternal}
```

	A		
B			C

Was hierbei auch interessant ist: die Skalierung von X - und Y -Koordinaten wird unabhängig voneinander getroffen, das heißt die Y -Koordinate eines Punktes als die X -Koordinate eines anderen zu verwenden funktioniert (meist) nicht ohne mathematische Operationen. Das Gitter wurde natürlich nachträglich hinzugefügt:

```
\draw[thin,xshift=0.5cm,yshift=0.5cm] (-1,-2) grid (3,2);
```

3.2.2 Rotation

Diese Definitionen befinden sich in der Datei: `source/Graphics/Tikz-Core/_LILLY_TIKZ_ROTATION`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LIB LILLYxGRAPHICS** geladen.

◊ `\rotateRPY[transform-point={0/0/0}]{roll}{pitch}{yaw}`

v1.0.4

Dieser Befehl wird verwendet um erstellte TikZ Grafiken zu drehen und dementsprechend anzupassen. Dieser Code entstammt der Feder von David Carlisle und Tom Bombadil^(a) und wird hier beispielhaft illustriert:

^(a)<https://tex.stackexchange.com/questions/67573/tikz-shift-and-rotate-in-3d>

```
\begin{tikzexternal}
\examplecube
\rotateRPY[-2/2/2]{13}{171}{55} %% Rotate set
\begin{scope}[draw=purple, text=purple,
            fill=purple, densely dashed, RPY]
\examplecube
\end{scope}
\draw[tealblue, ultra thick] (-2,2,2) -- (\savedx,2,2) -- (\savedx,
           \savedy,2) -- (\savedx,\savedy,\savedz) circle (0.25);
\rotateRPY[-2/-2/-2]{13}{171}{55}
\draw[Ao, ultra thick] (-2,-2,-2) -- (\savedx,-2,-2) -- (\_
           \savedx,\savedy,-2) -- (\savedx,\savedy,\savedz) circle (0.25);
\node at (2.5,-3.5){RPY: 13,171,55};
\end{tikzexternal}
```

RPY: 13,171,55

3.2.3 Automaten [WAR Work in Progress]

Diese Definitionen befinden sich in der Datei: `source/Graphics/Tikz-Core/_LILLY_TIKZ_AUTOMATEN`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxGRAPHICS` geladen.

Obwohl bereits TikZ eine Bibliothek für das Generieren von Automaten zur Verfügung stellt, wurde dieses (Work in Progress) Paket erstellt um darauf aufbauend schnell Automaten erstellen zu können. Der Grundbefehl lautet:

◊ `\loopTo[looseness=<1>]{arc}{node-name}{Text}{Orientierung}` v1.0.3

Dieser Befehl setzt grundlegend einen Pfeil, der von einem Knoten aus wieder zu sich selbst führt. Im folgenden sind 4 verschiedene Shortcuts, die für die klassischen Himmelsrichtungen die Pfeile vordefinieren:

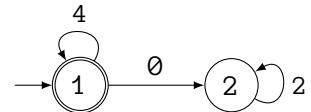
◊ `\loopTop[looseness=<1>]{node-name}{Text}, \loopRight[lsns=<1>]{node-name}{Text}, \loopLeft[lsns=<1>]{node-name}{Text}, \loopBot[lsns=<1>]{node-name}{Text}` v1.0.3

Im folgenden sei eine beispielhafte Verwendung gezeigt (der Automat muss keinen Sinn ergeben es soll lediglich die Nutzung verdeutlicht werden):

```

1 \begin{tikzternal}[scale=1,
2   every node/.style={minimum size=12pt,transform shape},
3   state/.style={circle, draw, minimum size=20pt},
4   every path/.style={draw, -latex},
5   every initial by arrow/.style={-latex, initial text=}]
6
7   \node[initial,accepting,state] (1) at (180:1){\texttt{\T{1}}};
8   \node[state] (2) at ( 0:1){\texttt{\T{2}}};
9
10  \draw (1) to node[pos=0.5,above,sloped]{\texttt{\T{0}}} (2);
11  \loopTop[4]{1}{\texttt{\T{4}}};
12  \loopRight[4]{2}{\texttt{\T{2}}};
13 \end{tikzternal}

```



Natürlich soll dieses Erstellen noch weiter stark vereinfacht werden. Des Weiteren wird darüber nachgedacht, einen akzeptierten Endzustand klarer zu markieren (Linien dicker, mehr Abstand etc.). Der Traum wäre, dass das Erstellen eines Automaten wie folgt funktioniert:

```

1 \begin{Automat}
2   \STATE[1]{180:1}{1};
3   \state[2]{0:1}{2};
4
5   \draw (1) to node[midway,above]{0} (2);
6
7   \loopTop[4]{1}{\texttt{\T{4}}};
8   \loopRight[4]{2}{\texttt{\T{2}}};
9 \end{Automat}

```

Die Befehle `\state` und `\STATE` sollen hierbei automatisch hochzählen können - pro Automat - aber über das optionale Argument lesbar einer Zahl zugewiesen werden. Die Umgebung Automat soll hierbei zusätzlich auch handhaben, dass automatisch alle Nodes mithilfe von `\T` geschrieben werden. Der entstehende Automat soll optisch identisch zum obigen sein, dies wird allerdings erst auf das Bedürfnis hin übernommen.

3.2.4 Schaltkreise [WAR Ausstehend]

3.2.5 Neuronen [WAR Work in Progress]

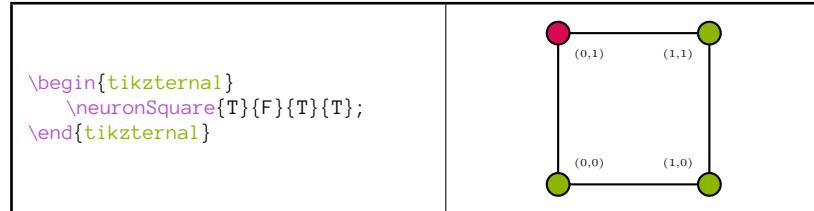
Diese Definitionen befinden sich in der Datei: `source/Graphics/Tikz-Core/_LILLY-TIKZ_NEURONS`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxGRAPHICS` geladen.

Da vor allem mit *Formale Grundlagen* der Wunsch danach aufkam, neuronale Netze schnell zu Texen, wurde dieses Paket entwickelt um das Paket mit den Schaltkreisen so zu erweitern, dass es erlaubt Perzepronen darin einzubauen, das Paket an sich befindet sich ebenfalls im Work in Progress-Status. *Das Schaltkreise-Paket ist ebenfalls noch nicht in LILLY integriert. Es befindet sich ebenfalls in einem Anfangsstadium und deswegen wird auch hierbei um Mithilfe bei der Weiterentwicklung gebeten.*

◊ `\neuronSquare{pos:00}{pos:01}{pos:10}{pos:11}`

v1.0.5

Es wurde bisher auch nur durch das Bereitstellen eines einzelnen Befehls implementiert: `\neuronSquare`. Dieser funktioniert seinerseits lediglich in einer `tikzpicture/tikzternal` Umgebung und zeichnet nichtmal ein Neuron, sondern lediglich die 2-D Repräsentation eines booleschen Raums, der wiedergibt unter welchen Eingabevektoren das Perzepron welchen Wert zurückliefert. Die 4 Parameter, die hierzu `\neuronSquare` benötigt, entsprechen der jeweiligen Binärdarstellung der Eingabevektoren. Eine beispielhafte Anwendung ist hier zu finden:

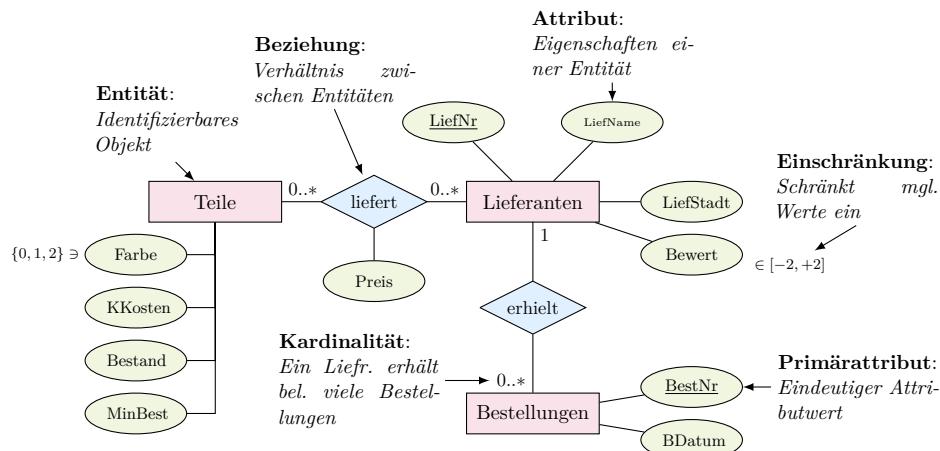


Hierbei steht ein T (true) natürlich für einen akzeptierten, ein F (false) entsprechend für einen nicht akzeptierten Befehl. Aktuell ist geplant, dass der Befehl auch für 1-, 3- und 4-dimensionale Räume eine Option anbietet (siehe für 4D: Titelgrafik *Grundlagen der Rechnerarchitektur*), die dann über einen einfacheren Namen abgegriffen werden kann. Weiter sollen dann *Formale Grundlagen* und *Grundlagen der Rechnerarchitektur* (boolesche Räume) diese Befehle nutzen anstelle der dafür eigens implementierten Grafiken. Weiter soll es möglich sein über ein optionales Argument die Position (relativ) zu bestimmen!

3.2.6 ER-Diagramme

Diese Definitionen liegen in der Datei `\LILLYxPATHxGRAPHICS/Tikz–Core/_LILLY_TIKZ_ER.tex`, sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxGRAPHICS` geladen.

Erstes Beispiel:



Erzeugt wurde dieses durch die folgenden Befehle:

◇ `\entity[Node Name]{Point}{Text}`

v1.0.9

Setzt eine Entität im Kontext eines Entity-Relationship-Diagramms. Wird kein expliziter `Node Name` angegeben, so wird er gleich dem `Text` gesetzt. Da dies natürlich nicht immer geht, bietet die Option hierfür einen Ausweg:

```
1 \begin{tikzpicture}
2   \entity{(0,0)}{Dieter};
3 \end{tikzpicture}
```

◊ `\relation[Node Name]{Point}{Text} [Width=<1>][Height=<0.5>]`

v1.0.9

Setzt eine Relation im Kontext eines Entity-Relationship-Diagramms. Wird kein expliziter Node Name angegeben, so wird er gleich dem Text gesetzt. Da dies natürlich nicht immer geht, bietet die Option hierfür einen Ausweg:

```
1 \begin{tikzpicture}
2   \relation{(0,0)}{Dieter};
3 \end{tikzpicture}
```

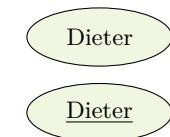


◊ `\attribute[Node Name]{Point}{Text},
\kattribute[Node Name]{Point}{Text}`

v1.0.9

Setzt ein Attribut beziehungsweise ein Schlüsselattribut im Kontext eines Entity-Relationship-Diagramms. Wird kein expliziter Node Name angegeben, so wird er gleich dem Text gesetzt. Da dies natürlich nicht immer geht, bietet die Option hierfür einen Ausweg:

```
1 \begin{tikzpicture}
2   \attribute{(0,1)}{Dieter};
3   \kattribute{(0,0)}{Dieter};
4 \end{tikzpicture}
```



3.2.7 Verzeichnis-Bäume [WAR Work in Progress]

Diese Definitionen liegen in der Datei `\LILLYxPATHxGRAPHICS/Tikz-Core/_LILLY_TIKZ_DIRECTORY.tex`, sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxGRAPHICS` geladen.

Dieses Paket ist bisher nur provisorisch und basiert auf <https://tex.stackexchange.com/questions/23647/drawing-a-directory-listing-a-la-the-tree-command-in-tikz#34268>, da `forest` noch ein paar Probleme macht.

◊ `env@directory[tikz args]`

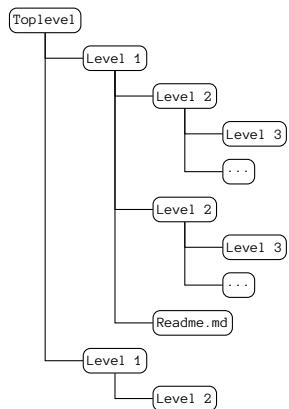
v2.0.0

Setzt ein Verzeichnis, beziehungsweise allgemein liefert diese Umgebung eine Möglichkeit relativ einfach etwas *Verzeichnisähnliches* zu setzen (ist ja, wie gesagt, noch in Arbeit ☺). Ein Beispiel:

```

1 \begin{directory}[scale=0.5]
2 \node{Toplevel}
3   child { node {Level 1}
4     child { node {Level 2}
5       child { node {Level 3} }
6       child { node {\dots} }
7     }
8     child { node {Level 2}
9       child { node {Level 3} }
10      child { node {\dots} }
11    }
12    child { node {Readme.md} }
13  }
14  child { node {Level 1}
15    child { node {Level 2} }
16  };
17 \end{directory}

```



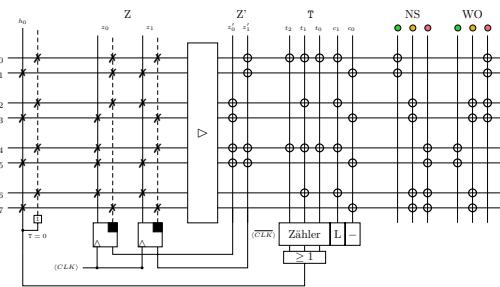
3.3 Mitgelieferte Grafiken

Diese Definitionen befinden sich in der Datei: `source/Graphics/LILLYxGRAPHICSxPROVIDER.sty`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxGRAPHICS** geladen.

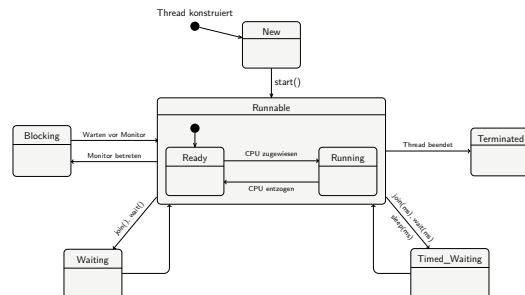
Dieser Teil existiert weiter auch als eigenes Paket mit: **LILLYxGRAPHICSxPROVIDER** und hängt vom Mutterpaket ab.

◊ `\getGraphics[width=<\linewidth>]{path}[height]` v2.0.0

Erlaubt den Zugriff auf zahlreiche Grafiken, die im Rahmen der Arbeit entstanden sind. Bei einer Angabe von Breite und Höhe gewinnt die Breite, da stets nur eine Dimension skaliert wird! Die bisher enthaltenen Grafiken können durch `jake get` abgerufen werden:

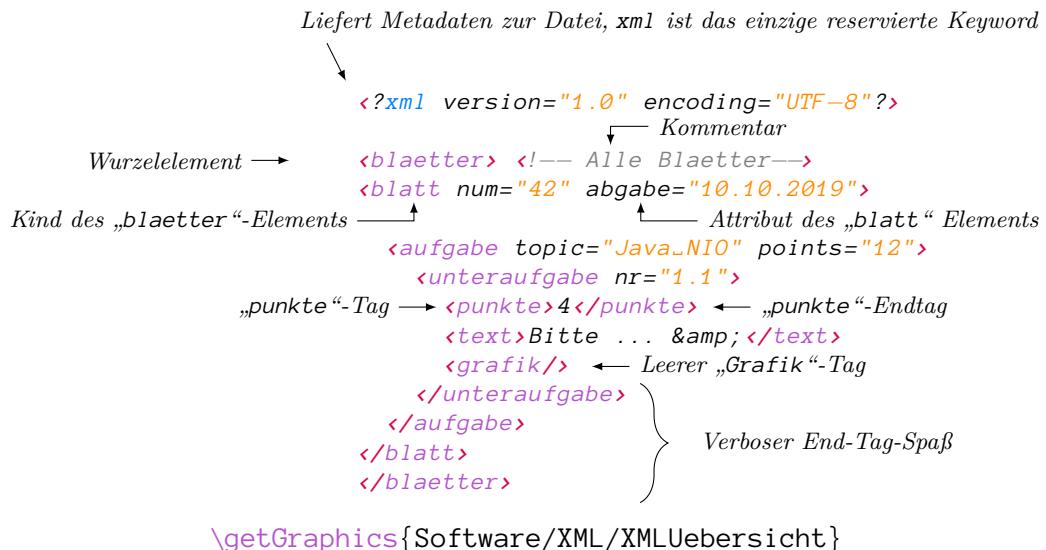


`\getGraphics{Rechner/PLAmpel}`



`\getGraphics{Software/ThreadState}`

Die Größe skaliert sich in der Regel automatisch, allerdings existieren auch Grafiken, die automatisch nicht skaliert werden, da sie Code oder andere nicht skalierfähige Elemente enthalten:

◊ `\getGraphicsPath{path}`

v2.0.0

Liefert den absoluten Pfad zu einer Grafik. Beispiel:

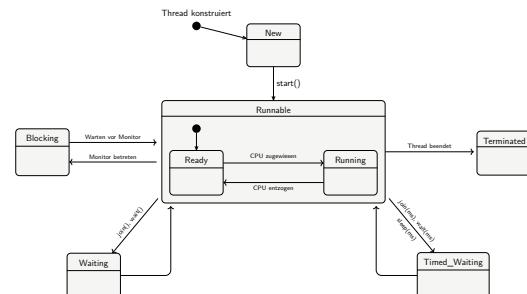
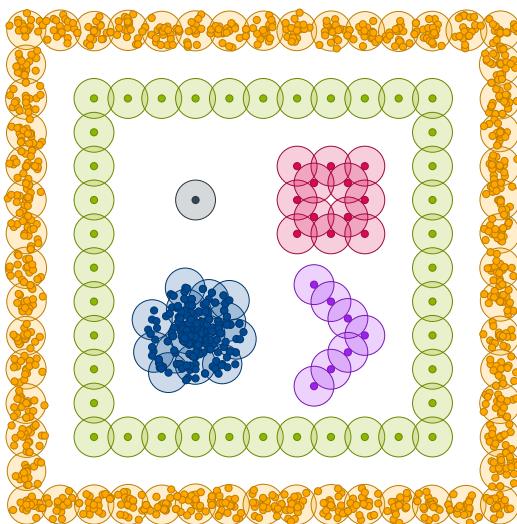
```
\getGraphicsPath{Software/XML/XMLUebersicht}
```

Liefert: source/Data/Graphics/Software/XML/XMLUebersicht.

◊ `\getPrerendered[width=\linewidth]{path}[height]`

v2.0.0

Erlaubt eine automatisch an die Seitenbreite skalierte Implementation von bereits vorberechneten Grafiken. Bei einer Angabe von Breite und Höhe gewinnt die Breite, da steht nur eine Dimension skaliert wird! Sie werden in der Grafiksammlung durch den Tag `pdf` gekennzeichnet (die Breite wurde im Beispiel angepasst, oder lassen sich durch das Anfügen eines „-pdf“-Suffix:



\getPrerendered{Eigene/Proseminar/Cluster/rolf-special}

\getPrerendered{Software/ThreadState-pdf}

Es gilt zu beachten, dass die bereits vorgenerierten Grafiken von den manuell generierten abweichen können!

3.4 Zusätzliche Optionen

3.4.1 Externalisierung

Diese Definitionen liegen in der Datei `\LILLYxPATHxGRAPHICS/Tikz-Core/_LILLY_TIKZ_GRAPHEN.tex`, sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxGRAPHICS** geladen.

Auf Basis von `environ` und der TikZ-Bibliothek `external` bietet diese Datei eine Möglichkeit Grafiken zu externalisieren. Wirklich aktiviert wird es allerdings nur auf Basis der *Jake*-Einstellung `lilly-external` (genau genommen läuft die Kontrolle über `\LILLYxEXTERNALIZE`), lagert dann allerdings alle Grafiken die durch folgende Umgebung definiert werden aus:

◊ `env@tikzternal[tikz args]`

v1.0.7

Wenn die Externalisierung (*Jake*-Einstellung `lilly-external`) aktiviert ist, wird die hierin enthaltene Grafik automatisch externalisiert. Sonst fungiert die Umgebung als normales `env@tikzpicture`.

3.4.2 Platzhalter

Diese Definitionen liegen in der Datei `\LILLYxPATHxGRAPHICS/Tikz-Core/_LILLY_TIKZ_PLATZHALTER.tex`, sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxGRAPHICS** geladen.

◊ `\imgplaceholder`

v1.0.1

Ein Platzhalter, der da eingebaut werden kann, wo ein Bild hingehört, allerdings noch keins ist ☺. So liefert `\imgplaceholder`:

Diese Grafik konnte aufgrund von Waffel-Mangel nicht realisiert werden.
Spende jetzt Waffeln um diese Grafik zu verwirklichen.

3.5 Weiterführende Symbole

3.5.1 Embleme

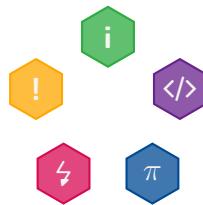
Diese Definitionen befinden sich in der Datei `source/Graphics/LILLYxEMBLEMS.sty`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxGRAPHICS** geladen.

Dieser Teil existiert weiter auch als eigenes Paket mit: **LILLYxEMBLEMS** und hängt vom Mutterpaket ab.

◊ `\infoEmblem, \warningEmblem, \errorEmblem, \mathEmblem, \codeEmblem`

v2.0.0

Hierbei handelt es sich um Shortcuts um einige Embleme direkt zu Setzen:



Hierbei bedienen sich die Befehle der Emblem-Definition \DefaultBaseEmblem.

◊ **\NewEmblem[*Emblem-Keys*][*Tikz-Args*]{*name*}**

v2.0.0

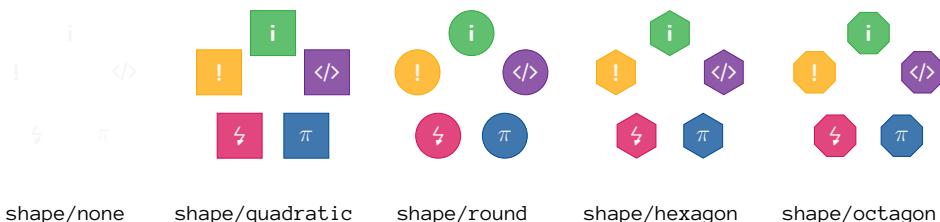
Definiert ein neues Emblem, wobei folgende Emblem-Keys zur Verfügung stehen, diese werden persistiert:

Bezeichner	Typ	Standard	Beschreibung
radius	<i>Length</i>	0.369cm	Radius des Symbols
shape	<i>Enum</i> ^(b)	shape/hexagon	Form des Hintergrunds
bgcolor	<i>Farbe</i>	DebianRed	Hintergrundsfarbe
bordercolor	<i>Farbe</i>	DebianRed	Rahmenfarbe
fgcolor	<i>Farbe</i>	MudWhite	Textfarbe
font	<i>Code</i>	\bfseries\large\sffamily	Schrift

So lassen sich relativ einfach GrundEmbleme definieren:

- 1 \NewEmblem[shape/none]{NoneEmblem}
- 2 \NewEmblem[shape/quadratic]{QuadraticEmblem}
- 3 \NewEmblem[shape/round]{RoundEmblem}
- 4 \NewEmblem[shape/hexagon]{HexagonEmblem}
- 5 \NewEmblem[shape/octagon]{OctagonEmblem}

So ist es zum Beispiel möglich durch die jeweilige Form das Aussehen der mitgelieferten Embleme zu modifizieren:



Das Erzeugen eines neuen Emblems mithilfe von \NewEmblem erzeugt einen neuen Befehl, entsprechend des Namens des Emblems. Der Befehl besitzt jeweils die folgende Signatur:

◊ **\<*name*>[*Tikz-Keys*]{*text*}**

v2.0.0

^(b) Allowed: none, quadratic, round, hexagon, octagon

So liefert zum Beispiel: `\OctagonEmblem{Hu}`: .

Oder: `\OctagonEmblem{\tiny stop}` .

Die Shortcuts von oben, wurden hierbei wie folgt definiert:

```

1 \gdef\infoEmblem{\, \DefaultBaseEmblem[draw=Leaf,fill=Leaf!75]{i}\,}
2 \gdef\warningEmblem{\, \DefaultBaseEmblem[draw=ChromeYellow,fill=>
    ChromeYellow!75]{!}\,}
3 \gdef\errorEmblem{\, \DefaultBaseEmblem[draw=DebianRed,fill=DebianRed!>
    75]{\wasysymLightning}\,}
4 \gdef\mathEmblem{\, \DefaultBaseEmblem[draw=DarkMidnightBlue,fill=>
    DarkMidnightBlue!75]{$\mathbf{\pi}$}\,}
5 \gdef\codeEmblem{\, \DefaultBaseEmblem[draw=DarkOrchid,fill=DarkOrchid!
    75]{\faCode}\,}

```

◊ `\textEmblem{Emblem}`

v2.0.0

Setzt ein Emblem für den Fließtext:  anstelle von . Die Argumentklammern können Vernachlässigt werden, das bedeutet es genügt das Schreiben von `\textEmblem\codeEmblem`.

◊ `\btextEmblem{Emblem}`

v2.0.0

Funktioniert identisch, setzt allerdings ein Emblem, welches die komplette Zeilenhöhe ausfüllt:



4

FARBEN

VIELE VIELE BUNTE FARBEN

VER 1.0.4

Damit die verwendeten Farben, je nach Profil und Wunsch in Paletten gruppiert gesetzt werden können, wurde dieses Paket ins Leben gerufen. Es befindet sich hier:

`\LILLYxPATHxDATA/Colors = source/Data/Colors`

Im Folgenden wird beschrieben wie grundlegend die Einbettung eines neuen Farbprofils ab VER 1.0.4 funktioniert. Bitte beachte, dass vor dieser Version ein Farbprofil noch alle Farben überschreiben und liefern musste, während seit dieser Version mit dem Überschreiben der Standard-Farben gearbeitet wird. Wichtig ist:

Jedes Farbprofil kann eigene Farben hinzufügen - hiervon wird aber stark abgeraten, da somit nicht mehr die Design-Unabhängigkeit von LILLY garantiert ist!

Bemerkung 7 – Standalone Color

Mit VER 2.0.0 wurde die Farben-Integration als eigenes Paket LIB LILLYxCOLOR etabliert, welches sich eigenständig über

`\usepackage{LILLYxCOLOR}`

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

4.1 Die normalen Farbprofile

Mit VER 2.0.0 werden die Hauptfarben generell mit diesem Paket zur Verfügung gestellt, während die Profile und Erweiterungen sich mit den Mappings befassen, dieser Prozess ist noch im Gange und natürlich wäre es Wünschenswert, wenn alle Farben über ein entsprechendes Mapping gesetzt werden.

Mit dem Paket LIB LILLYxLIST in VER 2.0.0, wurden die zur Verfügung stehenden Farben in Listen Organisiert:

- ◊ `\LISTxColors` (Quelle: LillyColorList)
- ◊ `\LISTxCompatColors` (Quelle: LillyCompatColorList)

Sie halten die jeweiligen Farben nach dem Schema: Name/R/G/B und können so entsprechend auch manipuliert werden. Die Farben können jeweils über folgenden Befehl Lilly gegenüber Registriert werden:

- ◊ `\registerColors{Liste:n/r/g/b}{Name}`, v2.0.0
`\updateColors{Liste:n/r/g/b}{Name}`

Dieser Befehl definiert die neuen Farben einmal mittels `\providecolor` (register) und

mit `\definecolor` (update). Die Listen-Signatur entspricht: Name der Farbe/R-Wert, /G-Wert/B-Wert. Da die Farben „nur“ registriert werden, kann man sie von außerhalb überschreiben, was allerdings zunichte gemacht wird, sofern man sie mittels `\updateColors` innerhalb des Dokuments überschreibt. Bisher sieht Lilly eine derartige Verwendung des Befehls nicht vor, er wird also intern nirgendwo verwendet.

In Lilly findet das registrieren der Farben wie folgt statt:

```
1 \storeLillyColorList{LISTxColors}
2 \registerColors{\LISTxColors}={}
3 \storeLillyCompatColorList{LISTxCompatColors}
4 \registerColors{\LISTxCompatColors}{Compat-}
```

Hier eine Auflistung der Standartfarben in `\LISTxColors`:

- | | |
|---|--|
| ● Butter (r: 255, g: 247, b: 155) | ● bondiBlue (r: 0, g: 149, b: 182) |
| ● Aureolin (r: 253, g: 238, b: 0) | ● antiVeg (r: 190, g: 238, b: 239) |
| ● Amber (r: 255, g: 191, b: 0) | ● DarkOrchid (r: 104, g: 34, b: 139) |
| ● ChromeYellow (r: 255, g: 167, b: 0) | ● Veronica (r: 160, g: 32, b: 240) |
| ● DarkChromeYellow (r: 255, g: 140, b: 0) | ● Orchid (r: 180, g: 82, b: 205) |
| ● Coquelicot (r: 255, g: 56, b: 0) | ● Amethyst (r: 153, g: 102, b: 204) |
| ● Cinnabar (r: 227, g: 66, b: 52) | ● AntiqueFuchsia (r: 145, g: 92, b: 131) |
| ● BrightMaroon (r: 195, g: 33, b: 72) | ● BritishRacingGreen (r: 0, g: 66, b: 37) |
| ● Cherry (r: 222, g: 49, b: 99) | ● DatmouthGreen (r: 0, g: 105, b: 62) |
| ● AlizarinCrimson (r: 227, g: 28, b: 54) | ● Ao (r: 0, g: 128, b: 0) |
| ● Amaranth (r: 229, g: 43, b: 80) | ● Leaf (r: 44, g: 171, b: 63) |
| ● AmericanRose (r: 255, g: 3, b: 62) | ● AppleGreen (r: 141, g: 182, b: 0) |
| ● Awesome (r: 255, g: 32, b: 82) | ● BrightGreen (r: 102, g: 255, b: 0) |
| ● BrightPink (r: 255, g: 0, b: 127) | ○ MudWhite (r: 245, g: 245, b: 243) |
| ● DebianRed (r: 215, g: 10, b: 83) | ○ LightGray (r: 224, g: 224, b: 224) |
| ● Crimson (r: 220, g: 20, b: 60) | ● AuroMetalSaurus (r: 110, g: 127, b: 128) |
| ● DarkMidnightBlue (r: 0, g: 74, b: 148) | ● Charcoal (r: 54, g: 69, b: 79) |
| ● Azure (r: 0, g: 127, b: 255) | |

Bemerkung 8 – Kompatibilität

Weiter gibt es die folgenden Farben, welche aus Kompatibilitätsgründen aus dem eagleStudiPackage übernommen wurden:

- | | |
|---|---|
| <ul style="list-style-type: none"> ● gold (<i>r: 255, g: 215, b: 50</i>) ● dgold (<i>r: 232, g: 177, b: 38</i>) ● mint (<i>r: 255, g: 128, b: 0</i>) ● dorange (<i>r: 255, g: 102, b: 0</i>) ● thered (<i>r: 255, g: 47, b: 47</i>) ● candypink (<i>r: 227, g: 112, b: 122</i>) ● ddpurple (<i>r: 128, g: 0, b: 128</i>) | <ul style="list-style-type: none"> ● beauty (<i>r: 104, g: 55, b: 107</i>) ● dpurple (<i>r: 86, g: 60, b: 92</i>) ● limegreen (<i>r: 51, g: 204, b: 51</i>) ● skyblue (<i>r: 60, g: 179, b: 113</i>) ● tealblue (<i>r: 51, g: 153, b: 255</i>) ○ superlightgray (<i>r: 240, g: 240, b: 240</i>) |
|---|---|

Sie sollten nicht mehr verwendet werden!

◊ \Hcolor, \HBColor

v1.0.9

Diese Farben können mithilfe von *Jake* auch durch den Parameter `lilly-signatur-farbe` gesetzt werden, wobei `\HBColor` immer eine etwas dunklere Variante der Farbe darstellt. Standartmäßig ist diese Farbe Leaf (●).

◊ \LillyxStorexCurrentColorProfile, \LillyxRestorexCurrentColorProfile

v2.0.0

Diese Befehle speichern das aktuelle Farbprofil und Laden es entsprechend wieder. Diese Mechanik wurde zum Beispiel hier verwendet um dynamisch die entsprechenden Farbprofile (wie das Druckprofil) anzuzeigen.

4.1.1 Das Standardfarbprofil

Diese Definitionen befinden sich in der Datei: `source/Data/Colors/_LILLY_DEFAULT_COLRPROFILE`. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB LILLYxGRAPHICS geladen.

◊ \LILLYxColorxInject

v1.0.1

Dieses Farbprofil wird nur geladen, wenn die Variable `\LILLYxColorxInject` nicht definiert ist.

Dieses Farbprofile definiert die Farben, welche LILLY für Links, Boxen usw. verwenden soll. Alle diese Befehle sollten auch bei eigenen Implementierungen und Erweiterungen angewendet werden, darum folgt hier eine Auflistung. Wichtig ist, dass mit VER 2.0.0 auch hier alle Farben jeweils in eine Liste geladen werden. Diese trägt den Namen `LillyProfileColors` (der Zugriff erfolgt wieder über: `\LISTxProfileColors`) und trägt die Verantwortung für die Konstruierten Farben. Lilly kümmert sich bisher noch nicht darum, dass nur gültige Farben in diese Liste gelangen, dies sollte allerdings nur eine untergeordnete Rolle spielen, da andere Farben schlicht ignoriert werden. Alle folgenden Farben werden durch das Präfix `LILLYxColorx` angeführt.

- | | |
|--|---|
| ● Definition (<i>DebianRed</i>) | ● Bemerkung (<i>Charcoal</i>) |
| ● Satz (<i>Ao</i>) | ● Zusammenfassung (<i>ChromeYellow</i>) |
| ● Beweis (<i>DarkMidnightBlue</i>) | ● Beispiel (<i>Aureolin</i>) |
| ● Lemma (<i>DarkMidnightBlue</i>) | ● Ubungsaufgabe (<i>Veronica</i>) |

- | | |
|--|--|
| <ul style="list-style-type: none"> ● Zusatzuebung (<i>Veronica</i>) ● LINKSxMainColor (<i>DebianRed!85!black</i>) ● LINKSxMainColorDarker (<i>DebianRed!75!black</i>) | <ul style="list-style-type: none"> ● LINKSxCiteColor (<i>DarkMidnightBlue</i>) ● LINKSxUrlColor (<i>DarkMidnightBlue</i>) ● TITLExCOLOR (<i>DebianRed!85!black</i>) |
|--|--|

Weiter gibt es noch die Farbe: `\LILLYxColorxLINKSxMainColorDarker` (●). Sie wird gemäß: `\LILLYxColorxLINKSxMainColor!90!black` generiert.

Beispielhaft lässt sich die Definitionsfarbe mit: `\LILLYxColorxDefinition` abfragen (●). Aus Flexibilitätsgründen wurden alle diese Farben als Befehle implementiert, um sie von den statischen Farben zu unterscheiden.

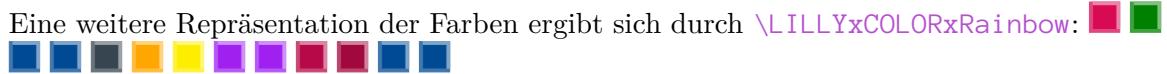
4.1.2 Das Druckprofil

Diese Definitionen befinden sich in der Datei: `source/Data/Colors/_LILLY_PRINT_COLRPROFILE`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxGRAPHICS` bereitgestellt und durch das Setzen des Druckmodus geladen.

Auch dieses Profil definiert seine Farben nur, wenn `\LILLYxColorxInject` nicht definiert ist! Die Präsentation der Farben erfolgt wieder mithilfe von: `\LISTxProfileColors`:

- | | |
|--|--|
| <ul style="list-style-type: none"> ● Definition (<i>DebianRed</i>) ● Satz (<i>Ao</i>) ● Beweis (<i>DarkMidnightBlue</i>) ● Lemma (<i>DarkMidnightBlue</i>) ● Bemerkung (<i>Charcoal</i>) ● Zusammenfassung (<i>ChromeYellow</i>) ● Beispiel (<i>Aureolin</i>) ● Uebungsaufgabe (<i>Veronica</i>) ● Zusatzuebung (<i>Veronica</i>) ● LINKSxMainColor (<i>DebianRed!85!black</i>) ● LINKSxMainColorDarker (<i>DebianRed!75!black</i>) ● LINKSxCiteColor (<i>DarkMidnightBlue</i>) ● LINKSxUrlColor (<i>DarkMidnightBlue</i>) ● TITLExCOLOR (<i>DebianRed!85!black</i>) | <ul style="list-style-type: none"> ● Definition (<i>DebianRed</i>) ● Satz (<i>Charcoal</i>) ● Beweis (<i>Charcoal</i>) ● Lemma (<i>Charcoal</i>) ● Bemerkung (<i>Charcoal</i>) ● Zusammenfassung (<i>ChromeYellow</i>) ● Beispiel (<i>Charcoal</i>) ● Uebungsaufgabe (<i>Charcoal</i>) ● Zusatzuebung (<i>Charcoal</i>) ● LINKSxMainColor (<i>Charcoal</i>) ● LINKSxMainColorDarker (<i>Charcoal!90!black</i>) ● LINKSxCiteColor (<i>Charcoal</i>) ● LINKSxUrlColor (<i>Charcoal</i>) ● TITLExCOLOR (<i>DebianRed</i>) |
|--|--|

Die Farbe `\LILLYxColorxLINKSxMainColorDarker` (●) wird hier mithilfe von: `\LILLYxColorxLINKSxMainColor!95!black` generiert.

Eine weitere Repräsentation der Farben ergibt sich durch `\LILLYxCOLORxRainbow`: 

4.2 Farberweiterungen

Es gibt eine Reihe an Farberweiterungen, die die oben definierten Druckprofile hinsichtlich einer gewissen Farbprägung ändern. Die von Lilly standardmäßig includierten Profile finden sich hier: `\LILLYxPATHxDATA/Colors/Extensions`:

Bezeichner	Farben
GREEN	
PURPLE	
CHESS	
VOID	

Die Farbprofile können durch das Setzen von `\LILLYxCOLORxEXTENSION` auf den jeweiligen Bezeichner geladen werden.

4.3 Weitere Planungen

- ◊ Elysium [WAR Ausstehend]
- ◊ Besseres Druckprofil [WAR Ausstehend]
- ◊ Weitere Farben [WAR Ausstehend] - Generische Farben wie „Rot“ auch als Befehl - zudem Lösung für Druckversion, sodass nirgendwo steht - der „Rote Kreis“ - wenn er dann eigentlich schwarz ist.

5

LISTINGS

IST THIS... THE MATRIX?

VER 1.0.0

Zum Setzen von Programmtexten innerhalb von Latexdokumenten stellt dieses Paket eine große Ansammlung verschiedener Sprachen und Dialekten zur Verfügung. Es befindet sich hier:

$$\text{\LILLYxPATHxLISTINGS} = \text{source}/\text{Listings}$$

Bemerkung 9 – Standalone Listings

Mit VER 2.0.0 wurde die Listings-Integration als eigenes Paket LIB LILLYxLISTINGS etabliert, welches sich eigenständig über

```
\usepackage{LILLYxLISTINGS}
```

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

*Sei es nun Formale Grundlagen, Einführung in die Informatik oder Grundlagen der Rechnerarchitektur, in jeder Vorlesungsreihe war es von Relevanz Quelltexte mit Syntax-Highlighting zu versehen. Hierfür verwendet LILLY die Bibliothek listings und fügt einige Styles und ein paar Sprachen hinzu, die ebenfalls frei gewählt werden können. Aktuell ist die Implementation an vielen Stellen noch weit weg von perfekt. So ist es in GDRA zum Beispiel immer noch vonnöten das Highlighting, von zum Beispiel addiu, mithilfe von *|\mipsADD* einzubinden. An einer Lösung hierfür wird aktuell gearbeitet, siehe weiter unten.*

5.1 Die grundlegenden Eigenschaften

5.1.1 Grundlegendes Design

Diese Definitionen befinden sich in der Datei: \LILLYxPATHxLISTINGS/LILLYxLISTINGS. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB LILLYxLISTINGS geladen.

Hier wird weiter von der Bibliothek LIB LILLYxLISTINGSxLANGUAGExCONTROL gebrauch gemacht, die sich mittels \RegisterLanguage um die Konstruktion der im Folgenden vermerkten Möglichkeiten kümmert!

Bemerkung 10 – Verwendetes Paket

LILLY verwendet nicht das normale listings-Paket, sondern greift auf das erweiterte Paket listingsutf8 zu, sofern dieses vorhanden ist. Es werden weiter Definitionen für alle Umlaute gesetzt, sowie eine Reihe an weiteren Ersetzungsregeln. Darunter fällt übrigens auch das Markieren von Zahlen. minted wird nicht verwendet um die Portabilität zu gewährleisten. Allerdings erlaubt \RegisterLanguage das Verwenden von minted.

Um dynamisch zu bleiben bindet LILLY nicht einfach verschiedene Stile ein, sondern Dateien, welche dann für sich definieren, welche Stile und Sprachen zusätzlich zur Verfügung stehen. Mithilfe von `\LILLYxListingsxLang` kann man das jeweilige Paket auswählen. Dieses Paket wird über den klassischen `\input{}`-Befehl eingebunden und zwar über folgende Anweisung:

```
\input{\LILLYxPATHxLISTINGS/Packages/_LILLY_PACK_\LILLYxListingsxPACK}
```

Standardmäßig wird so das MAIN-Paket geladen, welches alle hier definierten Sprachen mitliefert. Damit die zur Verfügung stehenden Sprachen auch verwaltet werden können, läuft die Verwaltung der Sprachen wieder über eine Liste. Die Liste *RegisteredLanguages* verwaltet hierbei die registrierten Sprachen (in der Signatur Sprache/Sprachbezeichner) und stellt für jede Sprache einen Shortcut zur Verfügung:

◊ `\c<Sprache> [Listing-Options] {Code}`

v1.0.9

Setzt den Code mit grauem Hintegrund. Zeilenumbrüche werden hier zwar durchgeführt, allerdings in der Regel nicht optimal gesetzt. Beispiel:

```
\cjava{public static void main(String[] args)}
```

Liefert: `public static void main(String[] args)`

◊ `\b<Sprache> [Listing-Options] {Code}`

v1.0.9

Setzt den Code farbig auf dem vorhandenen Hintegrund. Beispiel:

```
\bjava{public static void main(String[] args)}
```

Liefert: `public static void main(String[] args)`

◊ `\p<Sprache> [Listing-Options] {Code}`

v2.0.0

Setzt den Code im Präsentationsstil. Beispiel:

```
\pjava{public static void main(String[] args)}
```

Liefert: `public static void main(String[] args)`

◊ `\i<Sprache> [Listing-Options] {Code}`

v1.0.9

Lädt und setzt den Programmcode aus der entsprechenden Datei. Beispiel:

```
\ilatex[firstline=5,lastline=10]{Data/Listings.doc.tex}
```

Liefert:

```
1 \elable{chp:LISTINGS}\hypertarget{\LILLYxLISTINGS}Zum Setzen von ↵
    Programmtexten innerhalb von Latexdokumenten stellt dieses Paket ↵
    eine große Ansammlung verschiedener Sprachen und Dialekten zur ↵
    Verfügung. Es befindet sich hier:
2 \begin{center}
3   \blankcmd{\LILLYxPATHxLISTINGS} = \T{\LILLYxPATHxLISTINGS}
4 \end{center}
5
6 \begin{bemerkung}[Standalone Listings]
```

◊ env@<Sprache> [Listing-Options]

v1.0.9

Erlaubt das Setzen eines Textblocks in der jeweiligen Sprache, es ist die Kurzform von:

```
1 \begin{lstlisting}[style=\pgfkeysvalueof{/lillyxLISTINGS/globals/},
    listing style},language=<Sprache>]
2 Hier kann Code in der jeweiligen Sprache stehen.
3 \end{lstlisting}
```

Dieses Beispiel wurde zum Beispiel durch die Sprache `latex` gesetzt. Beispiel:

```
1 \begin{java}
2 public class SuperKlasse {
3     public static void main(String[] args) {
4         System.out.println("Hallo Welt");
5     }
6 }
7 \end{java}
```

Ergibt:

```
1 public class SuperKlasse {
2     public static void main(String[] args) {
3         System.out.println("Hallo Welt");
4     }
5 }
```

◊ env@<Sprache>*[Listing-Options]

v1.0.9

Entfernt die Zeilenummern eines sonst standardmäigen Listings:

```
1 \begin{java*}
2 public class SuperKlasse {
3     public static void main(String[] args) {
4         System.out.println("Hallo Welt");
5     }
6 }
7 \end{java*}
```

Ergibt:

```
public class SuperKlasse {
    public static void main(String[] args) {
        System.out.println("Hallo Welt");
    }
}
```

◊ env@plain<Sprache> [Listing-Options]

v2.0.0

Setzt ein Listing ohne irgendwelche zusätzlichen graphischen Hervorhebungen, außer sie werden durch die Optionen angegeben. Beispiel:

```

1 \begin{plainjava}
2 public class SuperKlasse {
3     public static void main(String[] args) {
4         System.out.println("Hallo Welt");
5     }
6 }
7 \end{plainjava}

```

Ergibt:

```

public class SuperKlasse {
    public static void main(String[] args) {
        System.out.println("Hallo Welt");
    }
}

```

◊ env@s<Sprache>

v2.0.0

Setzt ein Listing im Showcase-Design. Beispiel:

```

1 \begin{sjava}
2 public class SuperKlasse {
3     public static void main(String[] args) {
4         System.out.println("Hallo Welt");
5     }
6 }
7 \end{sjava}

```

Ergibt:

```

1 public class SuperKlasse {
2     public static void main(String[] args) {
3         System.out.println("Hallo Welt");
4     }
5 }

```

◊ \isLanguageLoaded{LanguageSignature}

v2.0.0

Prüft ob eine Sprache geladen ist. Als Argument wird hierbei die volle Sprachsignatur erwartet (Sprache/Sprachbezeichner) um auch doppelten Bezeichnern vorzubeugen.

◊ \isLanguageNameLoaded{LanguageName}

v2.0.0

Prüft ob eine Sprache geladen ist. Als Argument wird hierbei die volle Sprache erwartet, was doppelte Bezeichner natürlich ausschließt, allerdings in den meisten Fällen auch einfacher ist:

```

1 \isLanguageNameLoaded{java} % → TRUE
2 \isLanguageNameLoaded{waffel} % → FALSE

```

◊ `\lstshowcmd{command}`

v2.0.0

Kleiner Shortcut um auch den Inhalt eines Befehls als Listing zu setzen. Betrachte folgendes Beispiel:

```

1 \begin{multicols}{3}
2   \begin{ditemize}
3     \foreach \x in {public,static,void} {
4       \item \cjava{\x} vs. \lstshowcmd[language=1Java]{\x}
5     }
6   \end{ditemize}
7 \end{multicols}

```

Ergibt:

◊ `x` vs. **public**

◊ `x` vs. **static**

◊ `x` vs. **void**

◊ `\LILLYxwriteLst[1stArgs]{Code}`

v1.0.8

Setzt Programmcode entsprechend veralteter Definitionen.

Bemerkung 11 – Zugriff auf die eigentliche Sprachdefinition

Um keine Doppeldeutigkeit bezüglich der Sprachen zu erhalten werden alle LILLY-Sprachen durch das „1“-Prefix angeführt. So heißt es nicht „java“ sondern „1Java“, sofern die Sprache manuell geladen werden soll.

◊ `env@lstplain[1stArgs], env@lstonum[1stArgs]`

v1.0.9

Während erstere einfach nur Code ohne anderweitige Formatierungen setzt, entfernt letztere nur die Aufzählung entsprechender Zahlen:

```

1 \begin{lstplain}[language=1Java]
2 public static void main(String[] args) {
3   System.out.println("Hallo Welt");
4 }
5 \end{lstplain}
6 % Sowie:
7 \begin{lstonum}[language=1Java]
8 public static void main(String[] args) {
9   System.out.println("Hallo Welt");
10 }
11 \end{lstonum}

```

Ergibt:

```
public static void main(String[] args) {
  System.out.println("Hallo Welt");
}
```

Sowie:

```
public static void main(String[] args) {
    System.out.println("Hallo_Welt");
}
```

◊ `\LILLYxlstTypeWriter`

v1.0.0

Die allgemeine TypeWriter-Schriftart wird mithilfe von `\LILLYxlstTypeWriter` auf `AnonymousPro` gesetzt (*Sie wird auch hier für die Dokumentation verwendet*).

◊ `\LILLYxLISTINGSxFONTSIZE`, `\LILLYxLISTINGSxNUMxFONTSIZE`

v2.0.0

Setzen entsprechend die Schriftdaten für den Text und die Schriftdaten für die Zeilennummern in einer von `LILLYxLISTINGS`-Umgebung.

5.1.2 Das MAIN-Paket

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxLISTINGS/Packages/_LILLY_PACK_MAIN`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxLISTINGS` geladen.

Neben den geladenen Sprachen, liefert dieses Paket die Stildefinitionen die bereits in kleinen Teilen auch über PGF-Konfiguriert werden können. Dies wird aber wohl erst in zukünftigen Versionen sinnvoll konfigurierbar sein.

◊ `\lstcomment{text}`, `\lststring{text}`, `\lstnumber{text}`

v2.0.0

Setzt den Text so wie der Main-Stil den Code als Kommentar, String oder Zahl setzen würde. So kann auch durch `!* ... *` gesetzter Code korrekt formatiert werden:

```
1 \begin{java}
2 5 + 3 ergibt: !*\pgfmathparse{5+3}\pgfmathresult*!
3 5 + 3 ergibt: !*\lstnumber{\pgfmathparse{5+3}\pgfmathresult}*!
4 \isLanguageNameLoaded{java} // :yields: !*\isLanguageNameLoaded{java},
   *!
5 \isLanguageNameLoaded{java} // :yields: !*\lstcomment{,
   \isLanguageNameLoaded{java}}*!
6 "Im !*\LILLYxDOCUMENTxSUBNAME*! :D"
7 "Im !*\lststring{\LILLYxDOCUMENTxSUBNAME}*! :D"
8 \end{java}
```

Ergibt^(a):

```
1 5 + 3 ergibt: 8.0
2 5 + 3 ergibt: 8.0
3 \isLanguageNameLoaded{java} // → TRUE
4 \isLanguageNameLoaded{java} // → TRUE
5 "Im_Data/ Listings.doc :D"
6 "Im_Data/ Listings.doc :D"
```

^(a)Hier werden die Befehle nicht richtig markiert, da zum veranschaulichen von `\lststring` eine Sprache nötig war, die Zeichenketten als Datentyp besitzt.

◊ `\lstkwone{text}, \lstkwtwo{text}, ..., \lstkwsix{text}`

v2.0.0

Setzt den Text wie das entsprechende Keyword-Level:

```
1 \begin{java}
2 !* Hallo *! !* \lstkwone{Hallo} *! !* \lstkwtwo{Hallo} *!
3 !* \lstkwthree{Hallo} *! !* \lstkwfour{Hallo}* !
4 !* \lstkwfive{Hallo} *! !* \lstkwsix{Hallo} *!
5 \end{java}
```

Ergibt:

```
1 Hallo  Hallo  Hallo
2 Hallo  Hallo
3 Hallo  Hallo
```

Bemerkung 12 – Geladene Sprachen

Hier eine Auflistung aller Sprachen, die über das Main-Paket geladen werden:

◊ assembler	◊ latex	◊ sql	◊ haskell
◊ pseudo	◊ gepard	◊ xsl	◊ cpp
◊ mips	◊ java	◊ chr	◊ python
◊ bash	◊ xml	◊ prolog	◊ json

MAIN lädt noch das Paket MIPS, auf welches nun noch etwas weiter eingegangen wird...

5.1.3 Das MIPS-Paket

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxLISTINGS/Languages/_LILLY_LANG_MIPS`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LIB LILLYxLISTINGS** geladen.

Dieses Paket wurde vor allem im Rahmen von *Grundlagen der Rechnerarchitektur* erstellt und bindet das Paket `caption` mit ein, um die Positionierung von Titeln zu vereinfachen.

◊ `\gitRAW, \git` [WAR Veraltet]

v1.0.0

Fügen mithilfe von FontAwesome ein Github Symbol ein, welches auf ein Github-Repository verweist, indem sich alle in *Grundlagen der Rechnerarchitektur* verwendeten Codes befinden (https://www.github.com/EagleoutIce/MIPS_UniUlm_Examples/: ). Ursprünglich waren diese Definitionen nur für *Grundlagen der Rechnerarchitektur* gedacht und sollten auch schleunigst wieder dorthin verschwinden (TODO!).

Es werden einige weitere Stile definiert:

MIPS

Syntax-Highlighting für alle grundlegende MIPS-Befehle - verwendet 6 verschiedene Farben für verschiedene Arten von Keywords:

- Zeichenketten (*candyPink*)
- Befehle (*purple*)
- Register (*tealBlue*)
- Direktiven (*dgold*)
- Spezielle Befehle (*limeGreen*)
- Buzzwords (*thered*)
- Daten-Direktiven (*tealBlue!60!black*)

Weiter setzt es die Position der Zeilenummern auf die rechte Seite.

MIPSSNIP

Funktioniert analog zu MIPS, aber definiert das Design für kurze Ausschnitte.

Bemerkung 13 – MIPS

Das gesamte Mipspaket ist seit **VER 1.0.8** überholt und bedarf einiger Aufarbeitung, dennoch tut es seinen Dienst für die bisher existenten MIPS-Codes. Weitere Besonderheiten wie zum Beispiel Literates nebst der anfänglich implementierten stehen *nicht* zur Verfügung...

5.1.4 Kontrolle der Sprachen

Mit **VER 2.0.0** läuft die Registrierung einer Sprache über das Sub-Paket **LILLYxLISTINGSxLANGUAGExCONTROL** ab. Dieses definiert eine Menge an Befehlen, im Kern ist allerdings nur folgender von Relevanz:

◊ `\RegisterLanguage{Sprache}{1st-language}` **v2.0.0**
`[sig-list=<RegisteredLanguages>]`
`[name-list=<RegisteredLanguageNames>]`

Registriert eine Sprache wie java mit der entsprechenden Listing-Sprache `1st-language` wie `1Java`. Die Signatur der Sprache wird in die Liste `sig-list`, der ledigliche Name (wie `java`) in die Liste `name-list` eingetragen. Konstruiert werden die **weiter oben** beschriebenen Befehle für die Sprache. Erschaffen wir uns einmal die Sprache `rubberduck`:

```
1 \lstdefinelanguage{1Rubberduck}{  

2   comment=[1]{\#},  

3   morekeywords = {Quack, new},  

4   morekeywords = [2]{Duck}  

5 }
```

Bisher haben wir damit noch keine Sprache geladen oder irgenetwas Lilly-Kompatibles erzeugt:

```
\isLanguageNameLoaded{rubberduck} % → FALSE
```

Dies ändert sich durch folgenden Befehl:

```
\RegisterLanguage{rubberduck}{1Rubberduck}
```

Nun gilt die Sprache als geladen:

```
\isLanguageNameLoaded{rubberduck} % → TRUE
```

Und wir können sie im Code auch verwenden:

```

1 \begin{rubberduck}
2 Duck jens = new Duck(); # Eine neue Ente
3 Quack jens ::{
4     Quack Quack, Quack Quack
5     Quack Quack. # Entisch, es ist so simpel
6 }
7 \end{rubberduck}

```

Ergibt:

```

1 Duck jens = new Duck(); # Eine neue Ente
2 Quack jens ::{
3     Quack Quack, Quack Quack
4     Quack Quack. # Entisch, es ist so simpel
5 }

```

Analog existieren auch die inline-Befehle:

```
\prubberduck{Duck primus = new Duck();}
```

Ergibt: **Duck primus = new Duck();**

Wie die einzelnen Umgebungen heißen und wie sie dargestellt werden sollen lässt sich relativ frei konfigurieren. Für die durch `\lillylstset`-modifizierbaren Schlüssel (wie Präfix und Suffix des Befehls) steht die Dokumentation noch aus!

◊ `\LillyNewLstEnvironCore`{*Name*} {*Key*} {*In-Extra*} {*Out-Extra*} {*Language*}

v2.0.0

Dieser Befehl sollte nicht manuell aufgerufen werden, er wird aufgerufen und kann somit vom Nutzer modifiziert/überschrieben werden um die Eigenschaften der durch `\RegisterLanguage`-generierten Umgebungen zu modifizieren. Diese Befehl kümmert sich um die Standartumgebung wie `\begin{latex}`. Er erhält die entsprechenden Informationen über die jeweiligen Argumente. Die Standartdefinition dieses Befehls lautet ganz einfach:

```

1 \def\LillyNewLstEnvironCore#1#2#3#4#5{%
2   \lstnewenvironment{#1}[1][]{#3\lstset{##1}}{#4}
3 }

```

◊ `\LillyNewLstEnvironPlain`{*Name*} {*Key*} {*In-Extra*}

v2.0.0

Out-Extra} {*Language*}

Dieser Befehl sollte nicht manuell aufgerufen werden, er wird aufgerufen und kann somit vom Nutzer modifiziert/überschrieben werden um die Eigenschaften der durch `\RegisterLanguage`-generierten Umgebungen zu modifizieren. Diese Befehl kümmert sich um die Plain-Umgebung wie `\begin{plainlatex}`. Die Standartdefinition dieses Befehls lautet:

```

1 \def\LillyNewLstEnvironPlain#1#2#3#4#5{%
2   \lstnewenvironment{#1}[1][]{#3\lstset{xleftmargin=0pt,%
3     xrightmargin=0pt,%
4     numbers=none, numbersep=0pt, frame=none,%
4     rulecolor={}, backgroundcolor={}, ##1}}{#4}

```

5 }

- ◊ \LillyNewLstEnvironPresent{*Name*}{*Key*}{{*In-Extra*} {*Out-Extra*} {*Language*}}

v2.0.0

Dieser Befehl sollte nicht manuell aufgerufen werden, er wird aufgerufen und kann somit vom Nutzer modifiziert/überschrieben werden um die Eigenschaften der durch \RegisterLanguage-generierten Umgebungen zu modifizieren. Diese Befehl kümmert sich um die Presentation-Umgebung wie \begin{slate}. Die Standartdefinition dieses Befehls lautet:

```
1 \def\LillyNewLstEnvironPresent#1#2#3#4#5{%
2   \expandafter\xdef\csname#1\endcsname{\noexpand\leavevmode\noexpand\presentlst{#5}}
3   \expandafter\xdef\csname end#1\endcsname{\noexpand\endpresentlst}
4 }
```

- ◊ env@presentlst[*lst-args*][*tcb-args*]{*language*},
 env@plainlst[*lst-args*][*tcb-args*]{*language*},
 env@defaultlst[*lst-args*][*tcb-args*]{*language*}

v2.0.0

Liefert die Listings-Umgebungen jeweils als tcblisting. Beispiel:

```
1 \begin{defaultlst}{1Java}
2 System.out.println("Hallo Welt");
3 \end{defaultlst}
4 \begin{plainlst}{1Java}
5 System.out.println("Hallo Welt");
6 \end{plainlst}
7 \begin{presentlst}{1Java}
8 System.out.println("Hallo Welt");
9 \end{presentlst}
```

Ergibt:

```
1 System.out.println("Hallo_Welt");
```

sowie:

```
System.out.println("Hallo_Welt");
```

und:

```
1 | System.out.println("Hallo_Welt");
```

Letztere Box wird auch für den generierten Befehl verwendet. Die Inline-Befehle verwenden jeweils \LILLYxLSTINLINE, \LILLYxLSTBLANKINLINE, \LILLYxLSTINPL und \LILLYxLSTINLINEADVANCED.

5.2 Marker und weitere Befehle

5.2.1 Literates

Im Kontext verschiedener Programmiersprachen kam bald der Wunsch auf verschiedene Symbole entsprechend einfach Setzen zu können. Bisher werden alle diese Ersetzungsregeln über das Einbinden von **LILLYxLISTINGS** geladen und ermöglichen es, neben Umlauten auch Symbole einzubinden. Die Ersetzungsregeln werden nicht über eine Liste gehandhabt und sind ebenso vielfältig wie es die Bedürfnisse erfordern. Im Folgenden eine Auflistung aller in **VER 2.0.0** enthaltener Ersetzungsregeln:

:bs: „\“	:ws: „ “	:float: „f“	:bcmd: „\“
:bmath: „\$“	:cdots: „...“	:exp: „e“	:star: „*“
:emath: „\$“	:cdot: „·“	:yields: „→“	
:dollar: „\$“	:ldots: „...“	:lan: „⟨“	
:space: „ “	:c: „“	:ran: „⟩“	

5.2.2 Marker

Mit **VER 2.0.0** im Anfangsstadium befinden sich die jeweiligen Marker die es erlauben Fehler oder ganz Allgemein Code-stellen zu markieren, oder von Highlighting zu befreien:

```

1 \begin{java}
2 |info|import java.util.ArrayList;|info|
3
4 |plain|public class Example { |plain|
5     public static void main(String[] args) {
6         System.out.|err|PrintLn|err|("Hallo Welt");
7         if(|warn|args==null|warn|)
8             System.out.println("wau");
9     }
10 }
11 \end{java}
```

Ergibt:

```

1 import java.util.ArrayList;
2
3 public class Example {
4     public static void main(String[] args) {
5         System.out.Println("Hallo Welt");
6         if(args==null)
7             System.out.println("wau");
8     }
9 }
```

5.3 Advanced Listings

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxLISTINGS/LILLYxLISTINGSxADVANCED`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von `LILLYxLISTINGSxADVANCED` geladen.

◊ `\p<lang>{Code}`

v2.0.0

Setzt Analog zu `\c<lang>` den Code in einer Zeile im entsprechend Design. Hier allerdings ebenfalls das neue, modernere Design:

```
1 \pcpp{int main(int argc, char** argv)}
```

Ergibt: `int main(int argc, char** argv)`.

5.4 Runtimes

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxLISTINGS/LILLYxRUNTIMES`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von `LILLYxLISTINGSxADVANCED` geladen. Weiter existiert es als eigenständiges Paket `LILLYxRUNTIMES`.

Runtimes bieten die Möglichkeit Code einer Programmiersprache in Latex ausführen zu lassen und das Ergebnis ebenfalls im Latexdokument zu setzen. Hierfür wird eine bereits aufgesetzte Umgebung für die jeweilige Sprache benötigt, LILLY greift also auf einen bestehenden Compiler/Interpreter zurück. Alle mitgelieferten Runtimes befinden sich in der Liste `RegisteredRuntimes` und liefern:

◊ `\r<Runtime>[Mid-Text=<\, :>]{Code}`

v2.0.0

Führt den übergebenen Code in der jeweiligen Runtime aus und liefert das Ergebnis. So zum Beispiel mit `\rbash{ls . | tail -2}`: `ls . | tail -2`:

```
Lilly-Dokumentation.doc.upa  
README.md
```

`\rbash[liest:]{ls . | tail -4}`: `ls . | sort | tail -4` liefert:

```
Lilly-Dokumentation.doc.TOPIC  
Lilly-Dokumentation.doc.txt  
Lilly-Dokumentation.doc.upa  
README.md
```

◊ `\isRuntimeLoaded{runtimeName}`

v2.0.0

Testet analog zu `\isLanguageNameLoaded` ob eine entsprechende Runtime geladen ist:

```
1 \isRuntimeLoaded{bash} % → TRUE  
2 \isRuntimeLoaded{waffel} % → FALSE
```

Bemerkung 14 – Was es noch so gibt

Die Runtimes liefern, bisher noch nicht normiert, auch noch Befehle wie: `\preview-BashFile`, die eine bestehende Datei ausführen und das Ergebnis ausgeben. An einer Normierung und Erweiterung wird gearbeitet.

6

BOXEN

BOXES IN BOXES IN BOXES IN BOXES...

VER 1.0.0

Boxen aller Art werden durch dieses Paket generiert, welches verschiedene Optionen gibt:

```
\LILLYxPATHxCONTROLLERS = source/Controllers
```

Bemerkung 15 – Standalone Boxen

Mit VER 2.0.0 wurde die Listings-Integration als eigenes Paket LIB LILLYxBOXES etabliert, welches sich eigenständig über

```
\usepackage{LILLYxBOXES}
```

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

6.1 Grundlegendes

6.1.1 Eine kleine Einführung

Die 3 Standard-Designs, welche mit LILLY ausgeliefert werden lauten wie folgt:

DEFAULT	ALTERNATE	LIMERENCE
Satz 6.1 Nice Superwichtig	Satz 6.2 – Nice Superwichtig	Satz 6.3 – Nice Superwichtig

Mit VER 2.0.0 regelt Jake die jeweilige Variante und erlaubt es sogar, mehrere Boxmodi gleichzeitig generieren zu lassen:

```
jake <Datei> -lilly-boxes: "<Namen>"
```

Um eine Fassng für jede Box zu generieren entspräche das:

```
jake <Datei> -lilly-boxes: "DEFAULT_ALTERNATE_LIMERENCE"
```

wobei <Namen> mit einem der oben stehenden Bezeichner ersetzt wird. Die Bezeichner werden vom weiter unten näher beschriebenem Box-Controller wie folgt aufgelöst:

```
\userput{_LILLY_BOXES_}\LILLYxBOXxMODE% File
{\lillyPathData}% User Path
{\LILLYxPATHxDATA/POIs}% Lilly Path
```

Über genau dieses Verfahren lassen sich auch beliebig die Box-Designs erweitern.

6.1.2 Der Box-Controller

Diese Definitionen befinden sich in der Datei: Controllers/LILLYxCONTROLLERxBOX. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxBOXES** geladen. Sie befinden sich ebenfalls im eigenständigen Paket **LILLYxCONTROLLERxBOX**.

- ◊ **\LoadLillyBoxMode{*ModeName*}** **v2.0.0**
Lädt den Box-Modus mit dem Bezeichner *ModeName*. Dieser Befehl erlaubt es auch in einer Definition eines eigenen Designs ein anderes zu Laden, darunter ebenfalls ein bereits definiertes oder eigenes, es werden dieselben Pfade überprüft.
- ◊ **\LILLYxBOXxMODE** **v1.0.5**
Hierrüber wird ausgewählt welcher der jeweiligen Boxmodi verwendet werden soll. Standardmäßig wird dieser Befehl von *Jakc* gesetzt, aber natürlich kann dieser Befehl auch überschreiben werden.
- ◊ **\LILLYxBOXx<Bezeichner>xLock** **v1.0.8**
Enthält für die jeweilige Box, woran sich der Zähler orientieren soll. Enthält der Befehl TRUE, so wird ein Ungebundener Zähler verwendet. Wenn nicht definiert initialisiert durch:
- ◊ **\LILLYxBOXxHIGHLEVELxLOCK** **v1.0.8**
Enthält je nach Dokumenttyp entweder die höchste Hierarchie, an die ein Zähler gebunden werden kann oder TRUE. So erzeugt TRUE zum Beispiel den Zähler 4 und section in einem Dokument mit **\chapter: 1.13.4**.
- ◊ **\LILLYxBOXx<Bezeichner>xEnable** **v1.0.8**
Definiert, ob eine Box überhaupt angezeigt werden soll. Durch das Setzen auf FALSE, kann so eine Box aus dem Dokument genommen werden.
- ◊ **\LILLYxBOXx<Bezeichner>xBox** **v1.0.8**
Dieser Befehl besitzt (Stand **VER 2.0.0**) nicht für alle Boxbezeichner einen Effekt, steuert aber für bereits implementierte Boxen, ob diese durch das jeweilige Layout gesetzt werden sollen, oder ob die Box ohne die Box angezeigt werden soll. Die genaue Optik bestimmt wieder der jeweilige Modi.

Bemerkung 16 – Box-Kontrolle

Alle von Lilly generierten Boxen befinden sich mit **VER 2.0.0** in der Liste: RegisteredBoxes mit der Signatur Name/Bezeichner. So gehen die Konfigurationen wie folgt von statthen:

```
\def\LILLYxBOXxBeweisxBox{FALSE} % Deaktiviert Beweisboxen
```

Hier die definierten Umgebungen in ihrer freien Wildbahn und Gestalt:

Definition 6.1 – Titel

moin

```
1 \begin{definition}[Titel]
2     moin
3 \end{definition}
```



Definition 6.2 – Titel

moin

```
1 \begin{definition*}[Titel]
2     moin
3 \end{definition*}
```

Bemerkung 17 – Titel

moin

```
1 \begin{bemerkung}[Titel]
2     moin
3 \end{bemerkung}
```

Beispiel 6.1 – Titel

moin

```
1 \begin{beispiel}[Titel]
2     moin
3 \end{beispiel}
```

Satz 6.4 – Titel

moin

```
1 \begin{satz}[Titel]
2     moin
3 \end{satz}
```

Beweis 6.1 – Titel

moin

```
1 \begin{beweis}[Titel]
2     moin
3 \end{beweis}
```

Lemma 6.1 – Titel

moin

```
1 \begin{lemma}[Titel]
2     moin
3 \end{lemma}
```

Zusammenfassung 6.1 – Titel

moin

```
1 \begin{zusammenfassung}[Titel]
2     moin
3 \end{zusammenfassung}
```

Aufgabe 4 – Titel

moin

```
1 \begin{aufgabe}{Titel}{3}
2     moin
3 \end{aufgabe}
```

Nicht richtig darstellbar aber weiter existiert:

```
1 \begin{uebungsblatt}[Titel][2]
2     moin
3 \end{uebungsblatt}
```

◊ **env@task** [opt–Addons] {Titel} {Punkte} WAR Veraltet

v1.0.0

Ein aus dem eagleStudiPackage stammendes Relikt, welches nur aus Kompatibilitätsgründen gehalten wird. Ebenso:

◊ **\DEF** {Title} {Content}, **\BEM** {Titel} {Content}, ... WAR Veraltet

v1.0.0

Kompatibilitätsbefehle, der zur eagleStudiPackage-Zeit die Boxen gesetzt hat, nun allerdings die Daten an die jeweilige Umgebung weitergegeben.

◊ **\inputUB** {Name} {Nummer} {Pfad}, **\inputUBS** {Name} {Bezeichner} {Pfad}

v1.0.3

Binden eine Datei als Übungsblatt ein und erlaubt so, Übungsblätter in Mitschriften zu integrieren. Letzterer Befehl verwendet **env@uebungsblatt***, verändert also nicht die Nummer, was bedeutet, dass auch Buchstaben oder anderes als Bezeichner möglich ist. *Diese werden, entsprechend der Regel von env@uebungsblatt nur angezeigt, sofern \LILLYxMODExEXTRA den Wert TRUE enthält.*

Bemerkung 18 – Zugriff auf die Boxzähler

Der Zugriff auf die von Lilly unterhaltenen Boxzähler ist sehr Umständlich (Beispiel: `\thetcb@cnt@LILLYxBOXxDefinition`). Deswegen existiert das Hilfspaket `LILLYxBOXxCOUNTER`, welches hierfür Kurzbefehle definiert: `\CTRxDEF`, `\CTRxBEI`, `\CTRxBEM`, `\CTRxBESAT`, `\CTRxBEEW`, `\CTRxBELEM` und `\CTRxBEZSM`. So liefert `\arabic{\CTRxDEF} : 2.`

◊ `\RegisterBox[Box-Keys][Tikz-Keys]{Box-Name}{Title}{BoxID}`

v2.0.0

Registriert eine (neue oder alte) Box, die entsprechend Gesetzt werden kann. Die gezielte Verwendung dieses Befehls bedarf einiger Vorkenntnisse über das jeweilige Szenario. Es werden eine ganze Menge an Box-Keys gestattet, die auf einem anderen Verfahren persistiert werden (weswegen Box-gebundene Befehle mithilfe von `\noexpand` abgesichert werden müssen!). Alle so zur Verfügung stehenden Boxen werden durch `\RegisterBox` definiert.

Bezeichner	Typ	Standard	Beschreibung
name	<i>String</i>	noname	Name der Box
title	<i>String</i>	<name>	Titel der Box
boxcol	<i>Farbe</i>	black	Farbe für Links
preCode	<i>Code</i>		Befehle, die vor der Box gesetzt werden
inCode	<i>Code</i>		Befehle, die zu Beginn der Box gesetzt werden
outCode	<i>Code</i>		Befehle, die zu Ende der Box gesetzt werden
postCode	<i>Code</i>		Befehle, die nach der Box gesetzt werden
usestyle	<i>Box</i>	<Def>	Basisbox
emblem	<i>Code</i>		Titelverzierer
createlist	<i>Bool</i>	false	Erstelle Liste
customlist	<i>Bool</i>	false	Trage Box in Liste ein
boxenabled	<i>Bool</i>	true	Soll die Box angezeigt werden?
usebox	<i>Bool</i>	true	Box (true) oder das Plaindesign (false)?
lock	<i>Lock</i>	TRUE	Setzt die Zählersperre
listname	<i>String</i>	<name>	Name der Liste
listtext	<i>String</i>	<name>	Titel der Liste
listmen	<i>String</i>	NO	Mnemonic der Liste

Mit dem Registrieren einer Box, werden die folgenden Befehle für die jeweilige BoxID registriert. Sie werden expandiert, weswegen eine Absicherung mithilfe von `\noexpand` für

übergebene Befehle erfolgen sollte. Im Folgenden wird `<BoxID>` für die Befehle als Platzhalter verwendet:

- ◊ `\lillyxBOXx<BoxID>xName` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xTitle` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xBoxCol` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xPreCode` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xInCode` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xOutCode` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xPostCode` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xUseStyle` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xBoxEnabled` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xUseBox` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xEmblem` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xLock` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xListName` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xListItemText` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xListMen` v2.0.0

Sie speichern den Wert der sich jeweils vermuten lässt. Weiter werden noch die beiden Booleschen Werte gespeichert, die entsprechend es Zustands TRUE und FALSE:

- ◊ `\lillyxBOXx<BoxID>xCreateList` v2.0.0
- ◊ `\lillyxBOXx<BoxID>xCustomList` v2.0.0

Die Modifikation dieser ermöglicht eine weitere manuelle Anpassung der Box, allerdings wird von einem direkten, modifizierendem Zugriff abgeraten. `\RegisterBox` legt weiter auch noch die entsprechende Umgebung auf Basis des Namens an. Gilt es eine bestehende Box zu modifizieren, so gilt es folgenden Befehl zu nutzen:

- ◊ `\TransformBox[*][new-args]{OldBoxID}{Box-title}[new-name]{NewBoxID}` v2.0.0

Dieser Befehl nimmt die Werte einer bereits bestehenden Box und Transformiert sie in eine neue Box. Sollte keine Box mit der entsprechenden ID existieren, so wird die `BoxID` als Name angenommen auf dessen Namen die Umbenennung geschieht. So lassen sich auch bereits bestehende Umgebungen, die bisher keine Box dargestellt haben, entsprechend modifizieren und als Box rekreieren. Wird der Stern gesetzt, so wird die alte Box nicht gelöscht, sondern lediglich eine neue Box kreiert.

6.1.3 Die Boxmodi

Generell muss einer der unten aufgeführten Modi keine einzige Box definieren überladen oder modifizieren, das geladene Default-Paket wird jeweils für nicht überladene Boxen die Standartboxen zur Verfügung stellen. Folgende Boxbezeichner sind hierbei von Relevanz. Das Default-Design stellt hierbei mit `LILLYxLIST` die in der

Liste RegisteredBoxes aufgeführten Boxen zur Verfügung. An einer Normierung der aufgabe-Box wird gearbeitet: LILLYxBOXxDefinition, LILLYxBOXxBispiel, LILLYxBOXxBemerkung, LILLYxBOXxSatz, LILLYxBOXxBeweis, LILLYxBOXxLemma, LILLYxBOXxZusammenfassung, LILLYxBOXxAufgabe, LILLYxBOXxAufgabexPLAIN und LILLYxBOXxUebungsblatt. In der Regel wird weiter noch ein Design generiert, so setzt zum Beispiel das Default-Design: LillyxBOXxDesignxDefault.

Default-Design

Mit VER 1.0.0 stellt dieses Design den Urvater dar. Bis VER 1.0.6 überarbeitet hier die finale Form:

Zusammenfassung 6.2

Wichtige Box - Jeah

Wir haben schon viel gelernt, zum Beispiel, dass der Apfel nicht weit vom Stamm fällt. Das ist aber eigentlich dann auch schon so das Einzige, was es wirklich zu lernen gilt im Kontext dieser wundervollen Boxwelt!

Erzeugt durch den bekannten Aufruf:

```
1 \begin{zusammenfassung}[Wichtige Box – Jeah]
2   %% ....
3 \end{zusammenfassung}
```

Auf Basis des Pakets `tcolorbox` definiert LILLY das Design LillyxBOXxDesignxDefault mit folgender Implementation:

```
1 \tcbsset{LillyxBOXxDesignxDefault/.style={enhanced jigsaw,
2   pad before break=2mm, pad after break=2mm, %
3   lines before break=4, before skip=0pt, boxrule = 0mm, %
4   toprule=0.5mm, bottomtitle=0.5mm,bottomrule=1.2mm, %
5   after skip=0pt, enlarge top by=0.2\baselineskip, %
6   enlarge bottom by=0.2\baselineskip, %
7   sharp corners=south, enforce breakable}%
8 }
```

Auf Basis dessen werden nun die einzelnen Boxumgebungen generiert. Hier exemplarisch die obige Zusammenfassung:

```
1 \DeclareTCBox[auto counter]%
2   {LILLYxBOXxZusammenfassung}%
3   { O{} %% Title
4     O{Zusammenfassung \thetcboxcounter~} %% TitlePrefix
5     O{} %% tcb addonargs
6   }{%
7     LillyxBOXxDesignxDefault, %
8     colback=\LILLYxColorxZusammenfassung!5!white, %
9     colframe=\LILLYxColorxZusammenfassung, #3,%
10    title={\LILLYxDEFAUTxTYPESETxTITLE{\#1}{\#2}%
11      \ifx\LILLYxBOXxZusammenfassungxLock\true\%
12      \else\[-0.4\baselineskip]\fi}%
13    }% spacing
```

Hierbei verwendet das ganze Paket den vermerkten `\LILLYxDEFAULTxTYPESETxTITLE`, der selbst wie folgt konstruiert ist:

Bisher definiert LILLY die Counter über die Einstellung `auto counter` - dies soll aber bald auf das vom eagleStudiPackage Package verwendete `counter`-Verfahren umgestellt werden. Bis dato sieht eine exemplarische Definition einer Box wie folgt aus:

```

1 \DeclareTCBoxx[auto counter]%
2   {LILLYxBOXxDefinition}%
3   { O{} O{Definition \thetcboxcounter~} O{drop fuzzy shadow} }%
4   {LillyxBOXxDesignxDefault, colback=\LILLYxColorxDefinition!5!,%
5     white,%
6     colframe=\LILLYxColorxDefinition, #3,%
7     title={%
8       \begin{minipage}[t][\baselineskip][1]{\textwidth}%
9         \textbf{\textsc{\#2}} \hfill \textbf{\#1}%
10      \end{minipage}%
11    }%
12  }

```

Hiervon weichen nur 2 Definitionen ab. Die der Aufgaben-Box:

```

1 \DeclareTCBoxx{LILLYxBOXxAufgabe}{O{} O{} O{} }{enforce breakable,%
2   colback=white,colframe=black!50,boxrule=0.2mm,%
3   attach boxed title to top left={xshift=1cm,yshift*=1mm-,
4     \tcboxedtitleheight},%
5   varwidth boxed title*=3cm,%
6   boxed title style={
7     frame code={
8       \path [fill=white!30!black]%
9         ([yshift=-1mm,xshift=-1mm] frame.north west)%
10        arc [start angle=0,end angle=180,radius=1mm]%
11        ([yshift=-1mm,xshift=1mm] frame.north east)%
12        arc [start angle=180,end angle=0,radius=1mm];
13       \path [left color=white!40!black,right color=white!40!black,
14         middle color=white!55!black]
15         ([xshift=-2mm] frame.north west) -- ([xshift=2mm] frame.%
16           north east)%
17         [rounded corners=1mm] -- ([xshift=1mm,yshift=-1mm] frame.%
18           .north east)%
19         -- (frame.south east) -- (frame.south west)%
20         -- ([xshift=-1mm,yshift=-1mm] frame.north west)%
21         [sharp corners] -- cycle;%
22     },interior engine=empty,%
23   },
24   enhanced jigsaw, before skip=2mm,after skip=2mm,%
25   fonttitle=\bfseries, #3,%
26   title={#2 \ifthenelse{\equal{\#1}{} }{}{--\#1}}, %Aufgabe
27 }

```

Beispiel 6.2 – Wie man ein eigenes Box-Design erzeugt

Im Folgenden finden wir das Default-Design einer Deinition einfach hässlich und möchten es durch eine wunderschöne blaue Box ersetzen. Wir nennen das Design „quackbox“ und speichern es als `_LILLY_BOXES_quackbox.tex` im gleichen Ordner wie unser wundervolles Tex-Dokument. Weiter stört uns allgemein das Default-Design und wir wollen für unser Design gerne ALTERNATE als Grundlage nutzen. In die Definition unseres Boxdesigns schreiben wir also:

```

1 \LoadLillyBoxMode{ALTERNATE}% Laden des ALTERNATE-designs
2 \RenewTColorBox[use counter from=LILLYxBOXxDefinition]{%
3   LILLYxBOXxDefinition}%
4   { O{} O{} O{} }%
5   {#3, colframe=bondiBlue, title={#2#1},%
6 }
```

Das Tex-Dokument das wir nutzen möchten soll die article-Klasse und nicht Lilly nutzen, wir verwenden also nur die Bilbiothek `LIB LILLYxBOXES`. Die Nutzer-Definitionen sucht Lilly über den Pfad `\lillyPathData`. Wir schreiben also:

```

1 \documentclass{article}
2 \def\lillyPathData{./}% Suche im Dokumentordner
3 \def\LILLYxBOXxMODE{quackbox}%Nutze das gute Design
4 \usepackage{LILLYxBOXES}
5
6 \begin{document}
7   \begin{definition}[Hallo Welt]
8     Ich bin eine Definition
9   \end{definition}
10
11  \begin{bemerkung}[Hallo Welt]
12    Ich sehe wie eine Alternate-Bemerkung aus
13  \end{bemerkung}
14 \end{document}
```

Anstelle der Definition von `\LILLYxBOXxMODE` können wir, dank der Vordefinitionen in `LIB LILLYxVANILLA` folgendes schreiben:

```

1 \documentclass{article}
2 \usepackage{LILLYxBOXES}
3 \begin{document}
4   \LoadLillyBoxMode{quackbox}
5   \begin{definition}[Hallo Welt]
6     Ich bin eine Definition
7   \end{definition}
8
9   \begin{bemerkung}[Hallo Welt]
10    Ich sehe wie eine Alternate-Bemerkung aus
11  \end{bemerkung}
12 \end{document}
```

In beiden Fällen erzeugen wir die folgenden Boxen:

Definition 0.1 Hallo Welt

Ich bin eine Definition

Bemerkung 0.1 – Hallo Welt

Ich sehe wie eine Alternate-Bemerkung aus

6.1.4 Standart-Boxen

Das Paket LILLYxCONTROLLERxBOX definiert standardmäßig einige Boxen, die hier aufgeführt werden sollten, da sie in der Regel einfach so verwendet werden können:

◊ **env@definition[*Title*] [*TikZ-options*]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Definitionsbox im entsprechenden Boxmodi:

```
1 \begin{definition}[Hallo]
2   Hallo Welt!
3 \end{definition}
```

◊ **env@definition*[*Title*] [*TikZ-options*]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine hervorgehobene Definitionsbox im entsprechenden Boxmodi:

```
1 \begin{definition*}[Hallo]
2   Hallo Welt!
3 \end{definition*}
```

◊ **env@bemerkung[*Title*] [*TikZ-options*]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Bemerkungsbox im entsprechenden Boxmodi:

```
1 \begin{bemerkung}[Hallo]
2   Hallo Welt!
3 \end{bemerkung}
```

◊ **env@bemerkung*[*Title*] [*TikZ-options*]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Bemerkungsbox mit modifizierten Abständen (für Listen etc.) im entsprechenden Boxmodi:

```
1 \begin{bemerkung*}[Hallo]
2   \begin{itemize}

```

```

3   \item Hallo
4   \item Welt
5 \end{itemize}
6 \end{bemerkung*}
```

◊ **env@beispiel[Title][TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Beispielbox im entsprechenden Boxmodi:

```

1 \begin{beispiel}[Hallo]
2   Hallo Welt!
3 \end{beispiel}
```

◊ **env@beispiel*[Title][TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Beispielbox mit modifizierten Abständen (für Listen etc.) im entsprechenden Boxmodi:

```

1 \begin{beispiel*}[Hallo]
2   \begin{itemize}
3     \item Hallo
4     \item Welt
5   \end{itemize}
6 \end{beispiel*}
```

◊ **env@satz[Title][TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Satzbox im entsprechenden Boxmodi:

```

1 \begin{satz}[Hallo]
2   Hallo Welt!
3 \end{satz}
```

◊ **env@satz*[Title][TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Satzbox mit modifizierten Abständen (für Listen etc.) im entsprechenden Boxmodi:

```

1 \begin{satz*}[Hallo]
2   \begin{itemize}
3     \item Hallo
4     \item Welt
5   \end{itemize}
6 \end{satz*}
```

◊ **env@beweis[Title][TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Beweisbox im entsprechenden Boxmodi:

```

1 \begin{beweis}[Hallo]
2   Hallo Welt!
3 \end{beweis}
```

◊ **env@beweis*[Title] [TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Beweisbox mit modifizierten Abständen (für Listen etc.) im entsprechenden Boxmodi:

```

1 \begin{beweis*}[Hallo]
2   \begin{itemize}
3     \item Hallo
4     \item Welt
5   \end{itemize}
6 \end{beweis*}
```

◊ **env@lemma*[Title] [TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Lemmabox im entsprechenden Boxmodi:

```

1 \begin{lemma}[Hallo]
2   Hallo Welt!
3 \end{lemma}
```

◊ **env@lemma*[Title] [TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine hervorgehobene Lemmabox im entsprechenden Boxmodi:

```

1 \begin{lemma*}[Hallo]
2   \begin{itemize}
3     \item Hallo
4     \item Welt
5   \end{itemize}
6 \end{lemma*}
```

◊ **env@zusammenfassung*[Title] [TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Zusammenfassungsbox im entsprechenden Boxmodi:

```

1 \begin{zusammenfassung}[Hallo]
2   Hallo Welt!
3 \end{zusammenfassung}
```

◊ **env@zusammenfassung*[Title] [TikZ-options]**

v1.0.0

Wird durch `\RegisterBox` erzeugt und setzt eine Zusammenfassungsbox mit modifizierten Abständen (für Listen etc.) im entsprechenden Boxmodi:

```

1 \begin{zusammenfassung*}[Hallo]
2   \begin{itemize}
3     \item Hallo
4     \item Welt
5   \end{itemize}
```

```
6 \end{zusammenfassung}
```

◊ **env@aufgabe**[*TikZ-options*]{*Title*}{*Points*}

v1.0.0

Wird nicht durch `\RegisterBox` erzeugt und setzt eine Aufgabenbox im entsprechenden Boxmodi:

```
1 \begin{aufgabe}{hallo}{42}
2   Hallo Welt!
3 \end{aufgabe}
```

◊ **env@uebungsblatt**[*Title*] [*Number*] , **env@uebungsblatt***[*Title*] [*Number*]

v1.0.0

Wird nicht durch `\RegisterBox` erzeugt und setzt ein Übungsblatt, die Variante mit Stern setzt den aktuellen Aufgabenzähler nicht zurück was eine durchgängige Nummerierung ermöglicht. Siehe auch `\inputUB` und `\inputUBS`.

```
1 \begin{uebungsblatt}[Hallo]
2   Hallo Welt!
3 \end{uebungsblatt}
```

6.2 Info-Boxes

Diese Definitionen befinden sich in der Datei: `Controllers/LILLYxCONTROLLERxBOX`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von `LILLYxBOXES` geladen. Sie befinden sich ebenfalls im eigenständigen Paket `LILLYxBOXxINFOBOXES`.

Dieses Paket liefert gemeinsam mit `LILLYxEMBLEMS` einige nützliche Boxen wie:



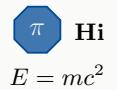
Dies ist eine Info-Box

Diese Info-Box kann ganz einfach mithilfe von folgendem Code erstellt werden:

```
1 \begin{infoBox}
2   Diese Info-Box kann ganz einfach mithilfe von folgendem Code ,
3   erstellt werden:
4 \end{infoBox}
```

Jeweils existiert auch noch eine mit einem Stern markierte Umgebung die sich entsprechend in die jeweilige Margin des Dokuments einnistet. Da die Dokumentation ohne große Margin für etwaige Paragraphen konzipiert wurde, lässt die Optik hier selbstredend zu Wünschen übrig. ^π Diese Box (inklusive klickbarem Marker im Text) wurde generiert durch:

```
1 \begin{mathBox*}{Hi}
2 $E=mc^2$
3 \end{mathBox*}
```



6.2.1 Wie es funktioniert

Gemeinsam mit `LILLYxMARGIN` und `LILLYxBOXxMARGIN` werden die Boxen für den Rand generiert, wobei diese sich für zweiseitige Dokumente selbstredend auch anpassen. Für die Boxen im Text wird `tcolorbox` verwendet. Eine neue Info-Box (inklusive „gestern“-Umgebung) definiert sich einfach durch folgenden Befehl:

◊ `\NewInfoBox[InfoBox-Keys][tcb-Keys]{Name}`

v2.0.0

Für die InfoBox Keys gibt es wieder eine ganze Liste an Feldern, die konfigurierbar sind. Sie werdenpersistiert, wobei das Präfix „`lillyxINFOBOXESx`“ verwendet wird:

Bezeichner	Typ	Standard	Beschreibung
<code>style</code>	<code>enum^(a)</code>	<code>style/limerence</code>	Design der Boxen
<code>bgcolor</code>	<code>Farbe</code>	MudWhite!75	Hintegrundsfarbe
<code>bordercolor</code>	<code>Farbe</code>	DebianRed	Rahmenfarbe
<code>fgcolor</code>	<code>Farbe</code>	Charcoal	Textfarbe
<code>titlefont</code>	<code>Code</code>	<code>\normalfont\bfseries</code>	Schrift des Titels
<code>textfont</code>	<code>Code</code>	<code>\normalfont</code>	Schrift des Textes

preCode	<i>Code</i>	Befehle, die vor der Box gesetzt werden
inCode	<i>Code</i>	Befehle, die zu Beginn der Box gesetzt werden
titleCode	<i>Code</i>	<long> ^(b)
outCode	<i>Code</i>	Befehle, die nach dem Titel gesetzt werden
postCode	<i>Code</i>	Befehle, die zu Ende der Box gesetzt werden
emblem	<i>Code</i>	Emblem, welches gesetzt werden soll
marker	<i>Code</i>	\textbf{!}
		Marker für Randnotiz

So wird zum Beispiel die `codeBox` wie folgt definiert:

```
1 \NewInfoBox[fgcolor={black},bgcolor={MudWhite!50},bordercolor={>
    DarkOrchid},emblem={\textEmblem\codeEmblem~},marker={\textbf{>
    faCode}}]{codeBox}
```

Es werden die folgenden Boxen vordefiniert: `infoBox*`^a, `warningBox*`[!], `errorBox*`^{\$}, `mathBox*`^π und `codeBox*`[▷]. Natürlich jeweils auch mit den normalen Umgebungen:

Ich bin eine Information

Nunc sed pede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum. Mauris metus. Curabitur lobortis. Curabitur sollicitudin hendrerit nunc. Donec ultrices lacus id ipsum.



Hi
Hallo Welt

Ich bin eine Warnung

Nunc sed pede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum. Mauris metus. Curabitur lobortis. Curabitur sollicitudin hendrerit nunc. Donec ultrices lacus id ipsum.



Hi
Hallo Welt

Ich bin ein Error

Nunc sed pede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum. Mauris metus. Curabitur lobortis. Curabitur sollicitudin hendrerit nunc. Donec ultrices lacus id ipsum.



Hi
Hallo Geld



Hi
Hallo Wält



Hi
Hello World

^(a) Allowed: none, limerence, framed

^(b) \leavevmode\smallskip\newline



Ich bin eine Mathe-Box

Nunc sed pede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum. Mauris metus. Curabitur lobortis. Curabitur sollicitudin hendrerit nunc. Donec ultrices lacus id ipsum.



Ich bin eine Code-Box

Pellentesque interdum sapien sed nulla. Proin tincidunt. Aliquam volutpat est vel massa. Sed dolor lacus, imperdiet non, ornare non, commodo eu, neque. Integer pretium semper justo. Proin risus. Nullam id quam. Nam neque. Duis vitae wisi ullamcorper diam congue ultricies. Quisque ligula. Mauris vehicula.

Weiter seien auch einmal die Stile veranschaulicht:



Warnung

Hallo Welt, na wie geht es dir? Ist super oder?
Jaaaaaa! Tiihihi.



Warnung

Hallo Welt, na wie geht es dir? Ist super oder?
Jaaaaaa! Tiihihi.



Warnung

Hallo Welt, na wie geht es dir? Ist super oder?
Jaaaaaa! Tiihihi.

style/none

style/framed

style/limerence

Sowie: !, ! und !.

◊ \dateBox{Datum/Text}

v2.0.0



Dieser Befehl ist einfach nur exemplarisch entstanden um zu demonstrieren, wie die Boxen auch misshandelt werden können ☺. natürlich sollte eigentlich \lillyxMarginxElement verwendet werden: \dateBox{\heute}. Wobei die Definition wie folgt stattfinden geht:

- 1 \NewInfoBox[fgcolor={DarkMidnightBlue}, bgcolor={MudWhite!0}, bordercolor={DarkOrchid}, emblem={\faCalendar~}, marker={}, style/none, titleCode={}]{@dateBox}
- 2 \def\dateBox#1{\begin{@dateBox*}{#1}\end{@dateBox*}}



Wie bereits erwähnt, werden die Einstellungen für eine InfoBox persistiert. Hierzu werden folgende Befehle verwendet:

- ◊ \lillyxINFOBOXESx<InfoBox>xDraw v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xBgColor v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xFgColor v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xBorderColor v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xTitleFont v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xTextFont v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xPreCode v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xInCode v2.0.0



- ◊ \lillyxINFOBOXESx<InfoBox>xTitleCode v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xOutCode v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xPostCode v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xEmblem v2.0.0
- ◊ \lillyxINFOBOXESx<InfoBox>xMarker v2.0.0

ALLZWECKMODULE UND KERN

EIN BUNTER TOPF VOLL NÜTZLICHES

VER 2.0.0

7.1 Die Allzweckmodule

Dieses Paket liegt hier:

```
\LILLYxPATHxUTIL = source/Util
```

Bemerkung 19 – Allzweckmodule standalone

Mit VER 2.0.0 wurden die Allzweckmodule als eigenes Paket LIB LILLYxUTIL etabliert, welches sich eigenständig über

```
\usepackage{LILLYxUTIL}
```

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

7.1.1 Lilly-Befehle

Diese Definitionen werden über die Bibliothek LIB LILLYxCOMMAND zur Verfügung gestellt. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB LILLYxUTIL geladen.

Die Befehle die hier zur Verfügung gestellt werden sollten nur dann verwendet werden, wenn es keine andere Möglichkeit gibt, das gewünschte Ergebnis zu erreichen.

◊ \LILLYcommand[*]{csname}[arg-count][defaultArg]{csdata} v1.0.0

Definiert einen Befehl analog zu \newcommand und \renewcommand, allerdings wird ignoriert, ob der Befehl davor bereits existiert. Es ähnelt damit einem \declarecommand, bietet aber die Gefahr, dass Pakete nichtmehr funktionieren, da ihre Befehle nicht so funktionieren wie erhofft.

◊ \gnewcommand[*]{csname}[arg-count][defaultArg]{csdata}, v1.0.4
\grenewcommand[*]{csname}[arg-count][defaultArg]{csdata}

Definiert die Befehle analog zu \long\def, macht also die definierten Befehle global verfügbar, behält aber jeweils die Eigenschaften von \newcommand und \renewcommand bei.

◊ \makerenewglobal, \makerenewlocal v2.0.0

Sorgen dafür, dass die zwischen den beiden Befehlen eingeschlossenen Environment-Definitionen global zur Verfügung gestellt werden. Ein Beispiel:

```
1 {
```

```

2  \makerenewglobal
3      \newenvironment{Waffel}{}{}
4  \makerenewlocal
5  \begin{Waffel}
6      Hi
7  \end{Waffel}
8 }
9 % Still available:
10 \begin{Waffel}
11     Hi
12 \end{Waffel}

```

◊ **\makeenvglobal{*environment*}**

v2.0.0

Konvertiert eine bestehende Umgebung in eine globale:

```

1 {
2     \newenvironment{Waffel}{}{}
3     \makeenvglobal{Waffel}
4     \begin{Waffel}
5         Hi
6     \end{Waffel}
7 }
8 % Still available:
9 \begin{Waffel}
10    Hi
11 \end{Waffel}

```

◊ **\providedef{*Name*}{{*Body*}}**

v2.0.0

Definiert den Befehl mit den Namen *Name*, sofern er noch nicht existiert mittels **\def**. Bisher ist die Angabe von Argumenten nicht vorgehsehen.

7.1.2 Kodierung

Diese Definitionen werden über die Bibliothek **LILLYxENCODING** zur Verfügung gestellt. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxUTIL** geladen.

Dieses Paket definiert und liefert keine zusätzlichen Befehle nebst denen der eingebundenen Pakete. Hier wird grundlegend nur alles geladen und so aufgesetzt wie es für ein Tex-Dokument gehört. Es werden die folgenden Pakete angefragt: **inputenc** (mit **utf8x**), **fontenc** (mit **T1**), **ngerman**, **textcomp**, **eurosym** und **microtype**. Weiter werden Umlaute für das Logfile gerendert. Das Paket kann auch dann geladen werden, wenn nicht alle diese, oder sogar gar keins dieser Pakete geladen ist, die erhoffte Wirkung bleibt allerdings aus.

7.1.3 Listen

Diese Definitionen werden über die Bibliothek **LILLYxLIST** zur Verfügung gestellt. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxUTIL** geladen.

Die durch diese Bibliothek bereitgestellten Listen sind bisher lediglich provisorisch und liefern die theoretischen Grundanforderungen die Lilly an eine Liste stellt.

- ◊ `\constructList[seperator]{ListName}` v2.0.0
Erstellt eine neue Liste, mit dem Präfix `lilly@list@`, die durch `\the<ListName>` abgefragt werden kann:
- ◊ `\the<ListName>` v2.0.0
Liefert die entsprechende Liste.
- ◊ `\delete<ListName>` v2.0.0
Löscht die Liste (löscht genau genommen nur die Inhalte der Liste).
- ◊ `\iter<ListName>` v2.0.0
Initiiert den Iterator für die jeweilige Liste.
- ◊ `\get<ListName>` v2.0.0
Speichert die Liste, vollexpandiert in `\lillyxlist`
- ◊ `\store<ListName>{name}` v2.0.0
Speichert die Liste vollexpandiert in `\<name>`.
- ◊ `\len<ListName>` v2.0.0
Liefert die Länge der Liste, sofern der Seperator ein Komma oder ein anderer von `\foreach` anerkannter Separator.
- ◊ `\containsList[ListName]{search}` v2.0.0
Vergleicht die vollexpandierten Werte einer *kommaseparierten* Liste mit dem entsprechenden Suchbegriff und expandiert zu `\true` (TRUE), wenn das Suchwort gefunden wird. Es ist geplant, dass dieser Ausdruck zu einem regulären TeX-if expandiert und in `\iflistcontains` umgenannt wird.
- ◊ `\typesetList[command=<typesetVoid>]{ListName}` v2.0.0
Setzt die Liste in deutscher Listen-Notation, wobei auf jedes Element das entsprechend als `command` übergebene Makro angewendet wird, welches als ein Argument den zu setzenden Text entgegen nehmen sollte. Der Standardbefehl `typesetVoid`, liefert den Text einfach wieder zurück, allerdings lässt sich wie folgt relativ einfach eine Liste an Befehlen setzen, der Befehl `\blankcmd` setzt den Text als Befehl in entsprechender Farbe (`\blankcmd{hallo}:\newline\hallo`):

```
\typesetList[blankcmd]{Ich, mag, züüge, so viele züüge}.
```

Liefert: `\Ich`, `\mag`, `\züüge` und `\so viele züüge`. Analog:

```
\newcommand{\meinBefehl}[1]{Jeah (\textit{#1})}
\typesetList[meinBefehl]{Ich, mag, züüge, so viele züüge}.
```

Liefert: Jeah (*Ich*), Jeah (*mag*), Jeah (*züüüge*) und Jeah (*so viele züüge*).

◊ `\setList{ListName}{Content}`

v2.0.0

Weißt der Liste `ListName` den Inhalt von `Content` zu.

◊ `\pusList{ListName}{Content}`

v2.0.0

Fügt der Liste `ListName` den Inhalt von `Content` zu, der Trenner wird automatisch so angefügt, dass kein überflüssiges Element entsteht:

```

1 \constructList[,] {Dieter}
2 \pushList{Dieter}{Hallo1}
3 \theDieter % → Hallo1
4 \pushList{Dieter}{Hallo2}
5 \theDieter % → Hallo1,Hallo2
6 \pushList{Dieter}{Hallo3}
7 \pushList{Dieter}{Hallo4}
8 \pushList{Dieter}{Hallo5}
9 \theDieter % → Hallo1,Hallo2,Hallo3,Hallo4,Hallo5

```

7.1.4 Zufall

Diese Definitionen werden über die Bibliothek `LILLYxRANDOM` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxUTIL` geladen.

Aktuell soll dieses Paket nur eine kleine Vereinfachung für alle mit dem Zufall im Verhältnis stehenden Operationen liefern.

◊ `\PickRandom{RndList}`

v2.0.0

Erstellt getreu der `pgf-randomlist`-Signatur eine Lite und wählt ein zufälliges Element:

```
\PickRandom{{Rot}{Blau}{Grün}{Gelb}{Orange}} % → Blau
```

◊ `\RandomInt[LowerBound=<0>]{UpperBound}`

v2.0.0

Liefert eine zufällige Zahl zwischen `LowerBound` und `UpperBound` (jeweils inklusiv):

```

\RandomInt{42} % → 2
\RandomInt[-42]{-1} % → -6

```

7.1.5 Kurzbefehle

Diese Definitionen werden über die Bibliothek `LILLYxSHORTCUTS` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxUTIL` geladen.

Bemerkung 20 – Verwendung der Kurzbefehle

Die Kurzbefehle die hier definiert werden existieren, auch wenn das dafür notwendige Paket nicht geladen ist. In diesem Fall sorgt die Verwendung für einen Fehler.

◊ `\lstfs[lineSpread]{fontsize}`

v2.0.0

Setzt die aktuelle Schriftgrößen Definition für `\LILLYxLISTINGS`, genau genommen werden `\LILLYxLISTINGSxFONTSIZE` sowie `\LILLYxLISTINGSxNUMxFONTSIZE` entsprechend definiert.

◊ `\T{Text}, \ltt`

v1.0.0

Der wohl am häufigsten verendete Kurzbefehl, setzt einen Text im Stil von `\LILLYxlstTypeWriter`. Analog hierzu setzt `\ltt` die aktuelle Schriftart auf die in `\LILLYxlstTypeWriter` definierte Schriftart.

◊ `\narrowitems, \closeritems`

v1.0.0

Manipulieren die Abstände in Listenumgebungen:

Ohne	<code>\narrowitems</code>	<code>\closeritems</code>
• Hallo	• Hallo	• Hallo
• Welt	• Welt	• Welt
• Wie geht es dir?	• Wie geht es dir?	• Wie geht es dir?

◊ `\tab[spacing=<1cm>]`

v1.0.0

An sich ein Wrapper für `\hspace*`, fügt einen horizontalen

Abstand ein.

◊ `\lreqn{PreText}{RightText}` [WAR Veraltet]

v1.0.0

Setzt Text an den linken (`PreText`) und entsprechend rechten (`RightText`) Rand der Zeile:

```
\lreqn{Hallo du da.}{Na wie gehts?}
```

Liefert:

Halло du да.

Na wie gehts?

Dieser Text wird auch dann gesetzt, wenn es die eigentliche Zeilenbreite verletzt.

So zum Beispiel: Hallo du da.

Na wie gehts?

◊ `\q[Text=<0>], \qq[Text=<0>]`

v1.0.1

Setzt den Text in einfachen (`\q`) und doppelten (`\qq`) Anführungszeichen: „hallo Welt“ und „Hallo Welt“.

◊ `\colvec{Vector}, \minicolvec{Vector}`

v1.0.0

Setzt einen Vektor:

$$\begin{aligned} \$\colvec{1\\3\\42}\$ \% &\rightarrow \begin{pmatrix} 1 \\ 3 \\ 42 \end{pmatrix} \\ \$\minicolvec{1\\3\\42}\$ \% &\rightarrow \left(\begin{array}{c} 1 \\ 3 \\ 42 \end{array} \right) \end{aligned}$$

- ◊ `\qedsymbol` v1.0.0
Setzt das Beweissymbol, sollte (um einfach geändert werden zu können) überall verwendet werden: `\qedsymbol`: ■
Wie zu sehen ist, wird das Symbol (standardmäßig) automatisch ganz nach rechts gesetzt.

- ◊ `\say{Text}` v1.0.0
Setzt analog zu `\qq` einen Text in Anführungszeichen:

```
\say{Hallo Welt} % → „Hallo Welt“
```

- ◊ `\engl[Surrounding]{Text}` v1.0.8
Setzt ein Wort als englische Übersetzung:

```
\engl{Hallo Welt} % → (engl. Hallo Welt)
\engl[x]{Hallo Welt} % → x(engl. Hallo Welt)x
```

- ◊ `\cd` v1.0.9
Kurzschreibweise für `\cdot`, funktioniert auch außerhalb von einer Matheumgebung: `\cd` ergibt: ..

- ◊ `\rom{Number}` v1.0.0
Konvertiert die übergebene positive Dezimalzahl in großgeschriebene römische Literale:

```
\rom{9} % → IX
\rom{-42} % → (fails, has to be positive!)
\rom{4096} % → MMMMXCVI
```

- ◊ `\fquad`, `\fqquad` v2.0.0
Setzt analog zu `\quad` und `\quad` Abstände, allerdings mit einer angepassten Möglichkeit den Abstand dynamisch anzupassen.

- ◊ `\ring{where}`, `\ringC[Color=<limegreen>]{where}`, `\bigRing{where}`, `\bigCRing[Color=<limegreen>]{where}` v1.0.0
Diese Befehle funktionieren nur in einer `env@tikzpicture`-entsprechenden Umgebung (wie `env@tikzternal`) und setzen entsprechend Kreise, im Folgenden wurden sie jeweils durch `\tikz` gesetzt:

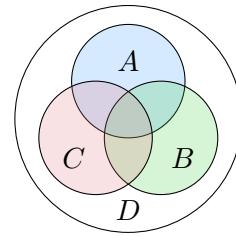
◊ <code>\ring{(0,0)}:</code> .	◊ <code>\bigRing{(0,0)}:</code> •
◊ <code>\ringC{(0,0)}:</code> .	◊ <code>\bigCRing{(0,0)}:</code> •

- ◊ `\firstcircle`, `\secondcircle`, `\thirdcircle`, `\bigcircle` WARN Veraltet v1.0.0
Setzt, Überraschung, Kreise an vordefinierten Positionen:

```

1 \begin{tikzpicture}[every node/.style={text=black,
2   opacity=1}]
3   \draw[fill=tealblue,fill opacity=0.2]
4     \firstcircle node[above] {\(A\)};
5   \draw[fill=limegreen,fill opacity=0.2]
6     \secondcircle node [below right] {\(B\)};
7   \draw[fill=candypink,fill opacity=0.2]
8     \thirdcircle node [below left] {\(C\)};
9   \draw \bigcircle node [yshift=-1.2cm] {\(D\)};
10 \end{tikzpicture}

```



Übrigens, diese Anzeige wurde durch ein `env@defaultlst` erzeugt, welches wie folgt eingerichtet wurde:

```

1 \begin{defaultlst}[] [listing side text,righthand width=3cm]{1Latex}
2 % ...
3 \end{defaultlst}

```

◊ `\dispnote[PreText]{Text}`

v1.0.8

Setzt eine `\parbox`, wird vor allem von `\note` und `\snote` verwendet um die Randformatierung zu setzen:

```

1 \dispnote{Hallo Welt, na wie geht es dir?}
2 \medskip\newline
3 \dispnote[Die Sonne: ]%
4   {Hallo Welt, na wie geht es dir?}

```

Hallo Welt,
na wie geht
es dir?
Die Sonne:
Hallo Welt,
na wie geht
es dir?

◊ `\note[PreText]{Text}, \snote[Text]{PreText}`

v1.0.8

Setzen die jeweilige Notiz in den Rand mithilfe von `\marginpar`, wobei `\RHD` und `\LHD` für die Pfeile verwendet werden. `\snote` unterscheidet sich in sofern, dass der Text in `\scriptsize` gesetzt wird:

```

1 \note[ABC]{Hallo Welt}
2 \snote[ABC]{Hallo Welt}

```

Gesetzt wurden sie hier Hallo Welt und hier Hallo Welt.

◀ ABC
◀ ABC

◊ `\nskip`

v1.0.9

Setzt den in der Länge `LILLYxNegativeSkip` definierten Abstand, der standartmäßig den Wert $-1.5\baselineskip$ hält.

◊ `\LILLYcoloredSQ{Color}`

v1.0.6

Zeigt die Farbe in einem formatierten Rechteck: `\LILLYcoloredSQ{bondiBlue}` ergibt: 

◊ `\LILLYxCOLORxRainbow`

v1.0.6

Zeigt alle Farben die im aktuellen Profil verwendet werden mithilfe von `\LILLYcoloredSQ` an: 

◊ `\fg`, `\gdra`, `\eidi`, `\la`, ...

v1.0.0

Setzt die entsprechenden Vorlesungen:

- ◊ `\fg`: *Formale Grundlagen*
- ◊ `\gdra`: *Grundlagen der Rechnerarchitektur*
- ◊ `\eidi`: *Einführung in die Informatik*
- ◊ `\la`: *Lineare Algebra*
- ◊ `\anaI`: *Analysis 1*
- ◊ `\pdp`: *Paradigmen der Programmierung*
- ◊ `\gdfs`: *Grundlagen der Betriebssysteme*
- ◊ `\pvs`: *Programmierung von Systemen*
- ◊ `\knn`: *Künstliche neuronale Netze*

◊ `\setLillyAuthor{new Author}`, `\setLillyAuthormail{new Authormail}`

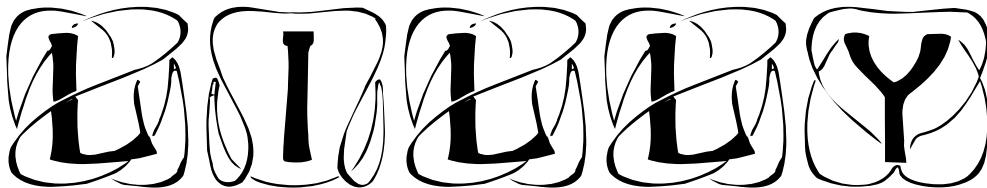
v2.0.0

Setzt die entsprechenden Felder `\AUTHOR` und `\AUTHORMAIL`, die eigentlich von `\JAKC` gesetzt werden neu.

◊ `\LillyLogo`

v2.0.0

Setzt das Lilly-Logo, welches sich auch in der Dokumentation wiederfindet:



Lilly is a Latex Lovable Yogurt

Die Formatierung verwendet übrigens *nicht* `\Acronym`, da sie möglichst ohne weiteres Paket funktionieren sollte.

7.1.6 Fallunterscheidungen

Diese Definitionen werden über die Bibliothek `\LILLYxSWITCHxCASE` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `\LILLYxUTIL` geladen.

Information, der Kerngedanke dieser Implementation basiert auf <https://tex.stackexchange.com/questions/64131/implementing-switch-cases>.

◊ `\case{Item}{if-matched}`, `\default{if-matched}`

v1.0.2

Funktionieren nur in der Umgebung `env@switch` und definieren analog zu den in Sprachen wie Java und C++ bekannten switch-case Unterscheidungen die Falldefinitionen für L^AT_EX.

◊ `env@switch{value}`

v1.0.2

Erlaubt die Definition einer Switch-Case Anweisung. Hier ein Auszug, wie sie auch in *Grundlagen der Rechnerarchitektur* verwendet wurde, um die Mnemonics aufzulösen:

```

1 \def\mnemonicDecode#1{%
2 \begin{switch}{#1}
3   \case{addi}{add~immediate}

```

```

4   \case{addu}{add~unsigned}
5   \case{sub}{subtract}
6   \case{addiu}{add~immediate~unsigned}
7   \case{subu}{subtract~unsigned}
8   \case{mult}{multiplicate}
9   \case{multu}{multiplicate~unsigned}
10  % ....
11  \default{#1}% should be last
12 \end{switch}
13 }
14 addi: \mnemonicDecode{addi} \\
15 multu: \mnemonicDecode{multu} \\
16 hallo: \mnemonicDecode{hallo}

```

Liefert:

addi: add immediate
 multu: multiplicate unsigned
 hallo: hallo

7.2 Der Kern

Dieses Paket liegt hier:

\LILLYxPATHxCORE = source/Core

Bemerkung 21 – Der Kern standalone

Mit **VER 2.0.0** wurden der Kern als eigenes Paket **LIB LILLYxCORE** etabliert, welches sich eigenständig über

\usepackage{LILLYxCORE}

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

Der Kern selbst definiert alle Pakete und damit Befehle, die so ziemlich von jedem Lilly-Paket (und wenn nur indirekt) verwendet werden. Es wird davon abgeraten die hier beschriebenen Pakete einzeln einzubinden, da sie ohnehin gegenseitige Abhängigkeiten aufweisen und somit nicht wirklich unabhängig voneinander sind! Weiter sind manche Pakete wirklich höchst minimalistisch, können allerdings in zukünftigen Versionen weiter ausgebaut werden!

◊ \LILLYxVERSION, \LILLYxSTATUS, \LILLYxVERSIONxLONG

v1.0.0

Setzt die aktuellen Daten zur Lilly-Version:

- ◊ \LILLYxVERSION: 2.0.0
- ◊ \LILLYxSTATUS: Work in Progress
- ◊ \LILLYxVERSIONxLONG: 2.0.0 - Jake ist auch nur Java

7.2.1 Booleans und Debug

Diese Definitionen werden über die Bibliotheken **LIB LILLYxBOOLEAN** und **LIB LILLYxDEBUG** zur Verfügung gestellt. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LIB LILLYxCORE**

geladen.

Da die beiden Bibliotheken so klein sind, werden sie hier gesammelt vorgestellt, wobei zuerst die einzigen in **LILLYxBOOLEAN** definierten Befehle gezeigt werden:

- ◊ **\true**, **\false**, **\n@true**, **\n@false** v1.0.0
Expandieren entsprechend zu TRUE und FALSE. Sie werden durch **\def** erzeugt, die durch **\newcommand*** generierten Pendants lauten **\n@true** und **\n@false**.
- ◊ **\LILLYxDEBUG** v2.0.0
Wird dieser Befehl auf **\true** gesetzt, aktiviert **LILLYxDEBUG** den folgenden Befehl:
- ◊ **\debugout{Output}** v2.0.0
Schreibt in den Log des Kompiliervorgangs, sofern der Debug aktiviert ist. Innerhalb des Dokuments kann dies durch **\lillydebugtrue** und entsprechend **\lillydebugfalse** aktiviert und wieder deaktiviert werden. Der Befehl **\LILLYxDEBUG** setzt hierbei lediglich den Initialwert.

7.2.2 Vanilla

Diese Definitionen werden über die Bibliothek **LILLYxVANILLA** zur Verfügung gestellt. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxCORE** geladen.

Dieses Paket definiert die wichtigsten, sonst von **Jakc** übernommenen Befehle, wenn **Jakc** nicht aktiv sein sollte (**\providetcommand**) und ermöglicht damit die ganzen standalone-Kompiliervorgänge ☺. Manche können von anderen Paketen, sofern **write18** aktiviert ist, noch korrigiert werden. Weiter werden Befehle wie **\LILLYxBOxxMODE** auf LIMERENCE und **\LILLYxPAPER** leer gesetzt, um auch anderen Paketen eine normale Arbeit zu ermöglichen.

- ◊ **\LILLYxCLSPATH**, **\LILLYxDOCSPATH** v1.0.4
Geben an, wo die **Lilly.cls** und das zu kompilierende Dokument liegen. Werden, wenn unbekannt einfach auf **./** gesetzt. Für diese Dokumentation halten sie die Werte:
 - ◊ **\LILLYxCLSPATH**: /home/eagle/texmf/tex/latex/Lilly
 - ◊ **\LILLYxDOCSPATH**: /eagle_extra/Studium/LILLY/Dokumentation
- ◊ **\LILLYxDOCUMENTNAME** v1.0.4
Name des Dokuments. Hier: Lilly-Dokumentation.doc.tex
- ◊ **\AUTHOR**, **\AUTHORMAIL** v1.0.4
Autor des Dokuments, siehe **\setLillyAuthor** und **\setLillyAuthormail**, sie enthalten die Daten zum Autor:
 - ◊ **\AUTHOR**: Florian Sihler
 - ◊ **\AUTHORMAIL**: florian.sihler@web.de
- ◊ **\LILLYxEXTERNALIZE**, **\LILLYxMODExEXTRA** v1.0.9

Sollen Grafiken mit `env@tikzternal` externalisiert werden, beziehungsweise sollen Übungsblätter und weitere Extras angezeigt werden? Standardmäßig werden sie, wenn nicht angegeben, auf FALSE gesetzt:

- ◊ `\LILLYxEXTERNALIZE`: FALSE
- ◊ `\LILLYxMODExEXTRA`: FALSE

◊ `\lillyPathLayout`, `\lillyPathConfig`, `\lillyPathData` v2.0.0

Enthalten jeweils die Pfade in denen nach weiteren Layouts, Konfigurationen oder Daten gesucht werden soll. Für ihre Auflösung wird `\userput` verwendet, sie werden also bevorzugt behandelt. Werden sie nicht angegeben, so werden sie auf das lokale Verzeichnis (`./`) gesetzt:

- ◊ `\lillyPathLayout`: source/Data/Layouts
- ◊ `\lillyPathConfig`:
- ◊ `\lillyPathData`:

◊ `\LILLYxFlavourText` v1.0.4

Hält einen eventuellen Flavour-Text. Dieser kann mit `LILLYxRANDOMxFLAVOURTEXT` auch generiert werden, ist aber sofern nicht gesetzt: „Kein Flavour-Text hinterlegt. Setze den Befehl LILLYxFlavourText entsprechend deines Wunsches.“

◊ `\LILLYxSemester` v1.0.0

Enthält das Semester, da nach der Vorlesung nicht gefragt ist, wenn es auf einem negativen Wert steht, wird es, sofern nicht angegeben, auf `-1` gesetzt^(a): 0.

7.2.3 Paket-Kontrolle

Diese Definitionen werden über die Bibliothek `LILLYxPACKAGExCTRL` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxCORE` geladen.

Die hier definierten Befehle `\LILLYxDemandPackage` und `\LILLYxLoadPackage` Werden weiter von Jake analysiert und gesammelt und können so

◊ `\LILLYxWANNABExERROR` v1.0.7

Ist `\false`, wenn alle angeforderten Pakete geladen werden konnten. Entnthalte sonst das letzte Paket, bei dem das gescheitert ist.

◊ `\LILLYxUSure[errorText=<Du hast mich ver...>]{Paket}` v1.0.7

Erwartet, dass das Paket geladen ist. Ist es das nicht, scheitert der Kompiliervorgang.

◊ `\LILLYxPoliteKnock[Endung=.sty]{Datei}{exists}{doesn't exist}` v1.0.7

Bindet die Datei nur ein, wenn sie existiert. Führt im positiven Fall den `exists`-Teil, sonst den `doesn't exist`-Teil aus.

◊ `\LILLYxDemandPackage{Paket}{Brief}{Error-Text}{Params}{Error}` v1.0.7

^(a) Damit kann auch mit `Jakc` unterschieden werden, der das Semester in diesem Fall bewusst auf `0` setzt.

Lädt das Paket mit dem Parametern `Params` und fügt die Beschreibung `Brief` hinzu. Wird es nicht gefunden, so wird der `Error`-Text ausgegeben und zusätzlich `Error` initiiert. Das Kompilieren scheitert, wenn das Paket nicht existiert. Beispiel:

```
%> %% https://ctan.org/pkg/amsfonts
\lillyxDemandPackage{amssymb}{Noch mehr Symbole}%% Package, Info
  {Wir wollen mehr Symbole}%% Error-Text
{}%%Params
{}
```

◊ `\lillyxLoadPackage{Paket}{Brief}{Error-Text}{Fallback}{Params}{Error}` v1.0.7

Lädt das Paket mit dem Parametern `Params` und fügt die Beschreibung `Brief` hinzu. Wird es nicht gefunden, so wird der `Error`-Text ausgegeben und zusätzlich `Error` initiiert. Das Kompilieren scheitert nicht, wenn das Paket nicht existiert, allerdings wird der `Fallback`-Code zusätzlich ausgeführt um so Befehle zu emulieren. Beispiel:

```
%> %% https://ctan.org/pkg/graphicx
\lillyxLoadPackage{graphicx}{Fuer tolle Grafiken}
  {Dieses Paket ist für includegraphics von noeten!}
  {\linput{\lillyxPATHxFALLBACKS/_LILLY_FALLBACK_GRAPHICX}}
{}{} % Sloppy includegraphics draft
```

7.2.4 Pfadverwaltung

Diese Definitionen werden über die Bibliothek `\lillyxPATH` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `\lillyxCORE` geladen.

◊ `\lillyxDOCUMENTxSUBNAME`

v1.0.5

Enthält den Namen des aktuellen Dokuments. Es gilt zu beachten, dass bewusst `\input` von dieser Regel ausgenommen ist. Ein anderweitig eingebundenes Dokument setzt diesen Befehl entsprechend. Aktuell trägt es den Wert: Data/AllzwecknCore.doc

◊ `\linput{Dokument}, \include{Dokument}, \linclude{Dokument}`

v1.0.5

Binden die Dokumente entsprechend ein, wobei die durch 1-angeführten Befehle vom definierten `\lillyxPATH` ausgehen.

7.2.5 Inhaltskontrolle

Diese Definitionen werden über die Bibliothek `\lillyxCONTROLLERxCONTENT` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `\lillyxCORE` geladen.

Dieses Paket versucht die korrekten Pfade für `\lillyxCLSPATH` und `\lillyxDOCSPATH` zu identifizieren, bisher ist dies die nur auf nicht-Windows-Systemen mit aktiviertem `write18` möglich. Allerdings kann durch das definieren folgenden Befehls die Pfadresolution deaktiviert werden, wenn sie sonst für Probleme sorgt. Der Befehl muss allerdings vor Lilly definiert werden:

◊ `\lillyxCMD`

v1.0.4

Ist dieser Befehl definiert, wird die automatische Pfadresolution von Lilly nicht durchgeführt!

- ◊ `\LILLYxPATHxROOT, \LILLYxPATHxFILExROOT` v1.0.4
Setzt das Wurzelverzeichnis für alle Lilly-Zugriffe. Wird für Lilly-Interna auf das lokale Verzeichnis gesetzt. Die Datei-Wurzel wird auf `source` gesetzt. Sie können beide von Jake modifiziert werden.
- ◊ `\LILLYxPATHx<Module>, \LILLYxABSPATHx<Module>` v1.0.4
Setzt für alle folgenden Module die jeweiligen Pfade, wobei `ABSPATH` jeweils den Absoluten Pfad hält. Beispiel:
 - ◊ `\LILLYxPATHxGRAPHICS:`
`source/Graphics`
 - ◊ `\LILLYxABSPATHxGRAPHICS:`
`/home/eagle/texmf/tex/latex/Lilly/source/Graphics`
 Alle entsprechenden Befehle: `\LILLYxPATHxCONTROLLERS` (`\LILLYxABSPATHxCONTROLLERS`), `\LILLYxPATHxDATA` (`\LILLYxABSPATHxDATA`), `\LILLYxPATHxGRAPHICS` (`\LILLYxABSPATHxGRAPHICS`), `\LILLYxPATHxFALLBACKS` (`\LILLYxABSPATHxFALLBACKS`), `\LILLYxPATHxHELPER` (`\LILLYxABSPATHxHELPER`), `\LILLYxPATHxLISTINGS` (`\LILLYxABSPATHxLISTINGS`), `\LILLYxPATHxMATHS` (`\LILLYxABSPATHxMATHS`), `\LILLYxPATHxBEAMER` (`\LILLYxABSPATHxBEAMER`), `\LILLYxPATHxUTIL` (`\LILLYxABSPATHxUTIL`), `\LILLYxPATHxCORE` (`\LILLYxABSPATHxCORE`) und `\LILLYxPATHxPRESENTER` (`\LILLYxABSPATHxPRESENTER`)
- ◊ `\LILLYxPATHxINDEX` v1.0.4
Pfad zur Index-Datei, ist aus historischen Gründen noch separat.
- ◊ `\lillyPathColorExtension` v1.0.4
Pfad zu den Farberweiterungen, ist aus historischen Gründen noch separat.
- ◊ `\dataInput{DataFile}` v1.0.4
Lädt eine Datei aus dem Datenverzeichnis.
- ◊ `\getSemester{Semester}{Vorlesung}` v1.0.4
Erhält den Pfad zu den Daten der entsprechenden Vorlesung.
- ◊ `\userput{File}{PriorityPath}{SecondaryPath}` v1.0.4
Versucht die Datei `File` zu Laden, wobei erst geschaut wird ob sie im `PriorityPath` existiert und dann, ob sie im `SecondaryPath` existiert. Lässt sich die Datei in beiden Pfaden nicht auffinden wird das Kompilieren nicht abgebrochen (dies schließt natürlich aus, wenn danach auf Befehle der jeweiligen Datei zurück gegriffen wird)!

7.3 Der Keyval-Parser

Diese Definitionen werden über die Bibliothek  `LILLYxKEYVALxPARSER` zur Verfügung gestellt und sind auch nur im Kontext der `Lilly.cls` gültig (oder genauer: sinnvoll).

Dieses Paket basiert auf dem Paket `koptions` und setzt alle folgenden Schlüssel in die

Familie LILLY mit dem Präfix LILLY@ (daher auch \LILLY@n etc.). Erstmal wird mittels \userput die Datei _LILLY_KEYVAL_GENERAL.tex in den Pfaden \lillyPathConfig beziehungsweise \LILLYxPATHxDATA/Configs geladen, die zusätzliche Optionen bereitstellen kann, die dann beim Laden von LILLY benutzt werden können. Die Standard-Version, die mit Lilly mitgeliefert wird, definiert die folgenden Optionen, primär für **LILLYxCONTROLLERxLAYOUT**, mehrere Optionen mit gleichem Effekt sind durch ein Komma getrennt:

Option	Beschreibung
debug	Aktiviert die Debug-Option für LILLYxDEBUG .
ElegantBook	Setzt das Design auf ELEGANT_BOOK .
PnP-Guide	Setzt das Design auf PNP_GUIDE .
Poems	Setzt das Design auf POEMS .
Paper	Setzt das Design auf PAPER .
Mitschrieb	Setzt das Design auf MITSCHRIEB .
Dokumentation	Setzt das Design auf Dokumentation (nicht dokumentiert bisher ☺).
Zusammenfassung, zsfg	Setzt das Design auf ZUSAMMENFASSUNG .
Uebungsblatt, ub	Setzt das Design auf UEBUNGSBLATT .

Wird keine der Optionen gewählt, so wählt LILLY das **PLAIN**-Design.

Natürlich gibt es noch eine ganze Menge an Möglichkeiten die, immer zur Verfügung stehen. Sie erwarten zum Teil ein zusätzliches Argument, dass bei nichtangabe einen Default-Wert erhält:

Option	Typ	Standart	Beschreibung
n	<i>String</i>	-1	Das wievielte Übungsblatt?
Semester	<i>String</i>	0	Das wievielte Semester?
Vorlesung	<i>String</i>	GDRA	Bezeichner der Vorlesung.
Typ	<i>String</i>	PLAIN	Zu verwendendes Layout.
Jake	<i>Boolean</i>	false	Jake -Unterstützung ⁽¹⁾ .
Universe	<i>Boolean</i>	false	Platzhalter.
paper	<i>Boolean</i>	false	Veraltet.
beamer	<i>Boolean</i>	false	Platzhalter.
beamerKiz	<i>Boolean</i>	true	Platzhalter.

⁽¹⁾In diesem Fall übernimmt **Jake** das Setzen von Vorlesung und Semester.

Der Typ hat weiterhin die Möglichkeit eigene Hooks zur Verfügung zu stellen, die vor der eigentlichen Arbeit von Lilly ausgeführt werden und so in der Lage sind zum Beispiel

das Dokumentformat zu ändern. Sie werden mittels `\userput` einmal in `\lillyPathConfig` und `\LILLYxPATHxDATA/Layouts/KeyvalHooks` gesucht, wobei der Name der Form `_LILLY_KEYVAL_<Typ>.tex` folgen muss, also zum Beispiel `_LILLY_KEYVAL_EIDI.tex`.

8

PRÄSENTATOREN

ALLES DARAN SETZEN, DIE DINGE HÜBSCH AUSSEHEN ZU LASSEN.

VER 2.0.0

Dieses Paket liegt hier:

`\LILLYxPATHxPRESENTER` = source/Presenter

Bemerkung 22 – Präsentatoren standalone

Mit VER 2.0.0 wurden die Präsentatoren als eigenes Paket LIB LILLYxPRESENTER etabliert, welches sich eigenständig über

```
\usepackage{LILLYxPRESENTER}
```

auch ohne das Verwenden der restlichen LILLY-Welt benutzen lässt.

8.1 Formatierungen

Diese Definitionen werden über die Bibliothek LIB LILLYxFORMATxCONTROL zur Verfügung gestellt. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB LILLYxPRESENTER geladen.

◊ \NoFormatChar

v2.0.0

Definiert das Zeichen, welches im Folgenden Befehl verwendet werden kann um eine spezifische Formtierung zu deaktivieren. Standardmäßig ist dies das Zeichen | (ein senkrechter Strich/Pipe-Symbol), lässt sich aber jederzeit ändern.

◊ \lilly@format@iterValue \@nil

v2.0.0

Liefert einen normalen Iterator, der über die durch Leerfelder getrennten Tokens eines Satzes iteriert. Für jedes Wort wird \lilly@format@step mit dem aktuell in \LILLY@FORMATTER@CURRENT gespeicherten Befehl aufgerufen. Dieser Aufruf separiert das jeweilige Wort mit dem ersten Buchstaben und überprüft die Anwendung des Befehls auf \NoFormatChar.

◊ \Acronym{Sentence}

v2.0.0

Formatiert den übergebenen Text so, dass jeder erste Buchstabe eines Wortes fett gedruckt wird. Hierbei wird der Befehl \TextBfFormat verwendet, der hier nicht dokumentiert ist, weil er an sich nur auf \textbf verweist und aus Lesbarkeitsgründen umbenannt wurde.

```
\Acronym{x Wie geht es ||dir?} % → x Wie geht es |dir?
```

Die Definition von \Acronym zeigt weiter, wie sich leicht durch die mitgelieferten Befehle ein ähnlicher Befehl definieren lässt:

```

1 \def\Acronym#1{%
2   \gdef\LILLY@FORMATTER@CURRENT{\TextBfFormat}%
3   \def\@Acronym{\lilly@format@iter#1 \@nil}{\ignorespaces\@Acronym}%
4 }

```

Allgemein genügt das Abändern von `\TextBfFormat` zu einem beliebigen anderen Befehl, der ein Argument akzeptiert. Dieser erhält dann jeweils das erste Zeichen des Wortes und kann es so formatieren wie es gewünscht ist.

◊ `\PoliteWords{Sentence}`

v2.0.0

Formatiert den übergebenen Text so, dass jeder erste Buchstabe eines Wortes groß gedruckt wird. Hierbei wird der Befehl `\UpperCaseFormat` verwendet, der hier nicht dokumentiert ist, weil er an sich nur auf `\MakeUppercase` verweist und aus Lesbarkeitsgründen umbenannt wurde.

```
\PoliteWords{x Wie geht |es ||dir?} % → X Wie Geht es |dir?
```

◊ `\ColorfulWords{Sentence}`

v2.0.0

Formatiert den übergebenen Text so, dass jeder erste Buchstabe eines Wortes farbig gedruckt wird. Hierbei wird der Befehl `\HighlightFormat` verwendet, der hier nicht dokumentiert ist, weil er an sich nur auf `\textcolor{\Hcolor}{#1}` verweist und aus Lesbarkeitsgründen umbenannt wurde.

```
\ColorfulWords{x Wie geht |es ||dir?} % → x Wie geht es |dir?
```

◊ `\doublealph{counter}`

v2.0.0

Während das durch L^AT_EX definierte `\alph` nur für Werte 1–26 definierte ist, definiert dieser Befehl die `\ifcase`-Anweisung für die Werte 1–702:

```

1 \newcounter{einzhler}
2 \setcounter{einzhler}{1}
3 \doublealph{einzhler} % → a
4 \setcounter{einzhler}{42}
5 \doublealph{einzhler} % → ap
6 \setcounter{einzhler}{702}
7 \doublealph{einzhler} % → zz
8 \doublealph{section} % → a
9 \doublealph{page} % → cb

```

8.2 Ornamente

Diese Definitionen werden über die Bibliothek `LILLYxORNAMENTS` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen.

Das folgende Paket definiert eine unglaubliche Anzahl an Befehlen, die nicht alle im Index aufgeführt werden.

◊ `\@@CreateOrnamentCommand{Number}{Name}{Defaultargs}` v2.0.0

Konstruiert einen Neuen Befehl der Struktur `\orna<Name>`, der auf das `\pgfornament` mit der entsprechenden Number abbildet. Weiter werden die Defaultargs übergeben, die ab da bei jedem Aufruf des Befehls Standardmäßig übergeben werden. Weiter registriert sich das Ornament in der durch LIB LILLYxLIST bereitgestellten Liste RegisteredOrnaments

◊ `\orna<Name>[ornaargs]` v2.0.0

Setzt das zuvor mit `\@@CreateOrnamentCommand` definierte Ornament. So zum Beispiel: `\ornalion` liefert .

Hier eine Auflistung aller Ornamente: rooster, cow, eagle, lobster, goat, fishes, elephant, horse, mice, jaguar, pig, peacock, pigeon, lion, crab, butterfly, owl, arroweagle, ox, fish, bird, crossleaf, spreadleaf, leafwall, simplebreak, leafbreak, spearbreak, wigglebreak, knight, rose, steeringwheel, handright, handleft, eye, archer, head, foot, clock, ship, shipleft, wing, scale, jug, hat, magichat, colorpalette, flag, feather, horseshoe, tree, shoe, leaf, branch, flowers, spadeflowers, tulips, elements, sword, posy, splinter, harp, angel, umberella und book

◊ `\OrnamentsBoxTitle[tikzargs]{Title}[width]` v2.0.0

Setzt eine Ornamentbox um den Title, wobei die Breite automatisch auf Basis der Länge des Titels generiert wird, sofern sie nicht durch width explizit angegeben wird.

```
1 \begin{center}
2   \OrnamentsBoxTitle{Hallo}
3 \end{center}
```



◊ `\OrnamentsUpper[Ornament Right][Ornament Left=<ornafeather>][tikz args]{Title}, \OrnamentsLower[tikz args]{TextRight}` v2.0.0

Setzt obere und untere Begrenzer, die auch in LIB LILLYxPOEMS verwendet werden.

```
1 \OrnamentsUpper{Meldung}
2
3 \OrnamentsLower{Wichtig!}
```

Ergibt:



Meldung



Information: Die Breite dieser Begrenzer passt sich an die aktuelle Zeilenbreite an, ist allerdings (bezüglich Liniendicke) auf Seiten im DIN A4-Format optimiert.

◊ \PresentAllOrnaments

v2.0.0

Erzeugt mithilfe von `LIB LILLYxTABLESxFOR EACH` eine tabellarische Auflistung aller Ornamente:

Die unterschiedlichen Höhen entstammen hierbei der Tatsache, dass dieser Befehl alle Ornamente (zur besseren Übersicht) mit derselben Breite setzt.

8.3 Tabellen

Diese Definitionen werden über die Bibliothek `LIB LILLYxTABLES` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxPRESENTER` geladen.

In diesem Paket werden einige neue Spaltentypen definiert, die Problemlos in so ziemlich jeder Tabelle verwendet werden können:

- b** Fettgedruckt (l)
- u** Mathematisch (c)
- g** Fußnotengröße (l)
- w** Fußnotengröße (X)
- i** Kursiv (l)
- L{#}** linskbündige m-Spalte der Breite #
- R{#}** rechtsbündige m-Spalte der Breite #

- C{#}** zentrierte m-Spalte der Breite #
- t** Spalte mit **\LILLYx1stTypeWriter** (l)
- T{#}** linksbündige p-Spalte der Breite # in **\LILLYx1stTypeWriter**-Schrift
- _** Abstand von 3em
- ~** Abstand von 1.5em
- .** Spaltenabstand von 1.5em

◊ \setrow{Commands}

v2.0.0

Setzt Befehle, die in dieser Zeile jeder Zelle vorangesetzt werden soll. Dieser Befehl lässt sich allerdings nicht einfach so verwenden, da ein derartiges Verhalten in Umgebungen wie **env@tabular** nicht vorgesehen ist. Jede Spalte auf die ein derartiger Effekt angewendet werden soll muss in der Definition mit einem ^ angeführt, die Spaltendefinition an sich wiederrum mit einem + abgeschlossen werden. Im Folgenden ein Beispiel:

```
1 \begin{tabular}{^1^c^r+}
2   Wir & sind & normal \\
3   \setrow{\itshape} Wir & sind & kursiv \\
4   Wir & sind & normal \\
5   \setrow{\bfseries} Wir & sind & fett
6 \end{tabular}
```

Wir	sind	normal
<i>Wir</i>	<i>sind</i>	<i>kursiv</i>
Wir	sind	normal
Wir	sind	fett

◊ \cleararrow

v2.0.0

Löscht den durch **\setrow** erzeugten Zeilenmodifikator:

```
1 \begin{tabular}{^1^c^r+}
2   Wir & sind & normal \\
3   \setrow{\itshape} Wir & sind & kursiv \\
4   Wir & sind & normal \\
5   \setrow{\bfseries} Wir & sind \cleararrow& fett
6 \end{tabular}
```

Wir	sind	normal
<i>Wir</i>	<i>sind</i>	<i>kursiv</i>
Wir	sind	normal
Wir	sind	fett

Dies wird durch + automatisch am Ende der Zeile angefügt.

◊ \headerarrow[*]

v2.0.0

Setzt mithilfe von **\setrow** eine Kopfzeile, der Stern behält die Grundlegende Formatierung der jeweiligen Zeile bei, sonst werden etwaige Formatierungen entfernt:

```
1 \begin{tabular}{^i^t^r+}
2   \headerarrow Wir & sind & normal \\
3   \headerarrow* Eine & tolle & Zeile \\
4           Super & tolle & Zeile
5 \end{tabular}
```

Wir	sind	normal
Eine	tolle	Zeile
<i>Super</i>	<i>tolle</i>	<i>Zeile</i>

- ◊ `\chaderrow, \normalrow, \smallrow, \footnotesizerow, \scriptsizerow, \tinyrow` v2.0.0

Setzt mithilfe von `\setrow` entsprechend formatierte Zeilen:

```
1 \begin{tabular}{^i^c^r+}
2   \chaderrow Eine & tolle & Zeile \\
3   \normalrow Eine & tolle & Zeile \\
4   \smallrow Eine & tolle & Zeile \\
5   \footnotesizerow Eine & tolle & Zeile \\
6   \scriptsizerow Eine & tolle & Zeile \\
7   \tinyrow Eine & tolle & Zeile
8 \end{tabular}
```

<i>Eine</i>	tolle	Zeile

8.3.1 Iterationen

Diese Definitionen werden über die Bibliothek `LILLYxTABLESxFOR EACH` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen.

- ◊ `\tabadd{tokens}, \etabadd{tokens}` v2.0.0

Fügt die `tokens` dem aktuellen Speicher (`\@@tabular@tokens`) hinzu. Zu beachten ist, dass `\etabadd` die `Tokens` mittels `\edef` expandiert.

- ◊ `\tabreset` v2.0.0

Leert den durch `\tabadd/\etabadd` erzeugten Speicher. Sollte, um sicher zu gehen, vor jeder neuen Iteration getätigkt werden.

- ◊ `\tabprint` v2.0.0

Gibt die aktuellen Tokens aus. Hier ein Beispiel:

```
1 \tabreset%
2 \foreach \i in {1,...,5} {
3   \etabadd{Zeile \i}%
4   \tabadd{ & Hallo Welt \\}%
5 }%
6 \begin{tabular}{ll}
7   \tabprint
8 \end{tabular}
```

Zeile 1	Hallo Welt
Zeile 2	Hallo Welt
Zeile 3	Hallo Welt
Zeile 4	Hallo Welt
Zeile 5	Hallo Welt

- ◊ `\tabforeach{Var}{Elements}{Body}` v2.0.0

Vereinfachte Möglichkeit um eine Tabelle so zu generieren.

```
1 \tabforeach{\i}{1,...,5}{%
2   \etabadd{Zeile \i \amp Hallo Welt}
3 }
4 \begin{tabular}{ll}
5   \tabprint
6 \end{tabular}
```

Zeile 1	Hallo Welt
Zeile 2	Hallo Welt
Zeile 3	Hallo Welt
Zeile 4	Hallo Welt
Zeile 5	Hallo Welt

Der Befehl `\amp` steht nur innerhalb von `\tabforeach` zur Verfügung und erlaubt so das setzen von einem & innerhalb von `\etabadd`, was es hier ermöglicht den weiteren Aufruf zu `\tabadd` einspahrt. Das Zeilenende wird automatisch am Ende jeder Iteration angefügt, `\tabreset` wird automatisch aufgerufen.

8.3.2 Weitere Designs

Diese Definitionen werden über die Bibliothek **LILLYxTABLESxMATERIAL** zur Verfügung gestellt. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxPRESENTER** geladen.

Bemerkung 23 – Das tabu-Paket und seine Leiden

Die Bibliothek **LILLYxTABLESxMATERIAL** wurde erst auf Basis des tabu-Pakets (<https://www.ctan.org/pkg/tabu>), da das Paket seit 2011 nichtmehr aktiv entwickelt wird unkluell ohne Maintainer vor sich hin vegetiert, ist es möglich, dass die durch Lilly etablierten Fixes und Variationen dennoch nicht funktionieren, was manche der folgenden Tabellen nicht „gut“ aussehen lässt oder sogar unbrauchbar macht.

- ◊ `env@mtable[*][Headerrow=<\MHeaderRow>][PreCode]{Header}[FirstRow=<MudWhite!10>][Second Row=<MudWhite!90>]`

v2.0.0

Setzt eine Tabelle in einem an das Material-Design des Editor Atom angelehnten Design:

```
1 \begin{mtable}{11}
2   Hallo & Welt \\
3   Na & Wie \\
4   geht & es \\
5   dir & denn?
6 \end{mtable}
```

Hallo	Welt
Na	Wie
geht	es
dir	denn?

Die Variante mit einem Stern, besitzt keine explizite Formatierung der Kopfzeile.

- ◊ `env@mtable[*][Headerrow=<\MHeaderRow>][PreCode]{Header}[Mulitpage Header][FirstRow=<MudWhite!10>][Second Row=<MudWhite!90>][Next Headerrow=<\MNHeaderRow>]`

v2.0.0

Setzt eine Tabelle, die gut und gerne über mehrere Seiten gehen kann. Die Kopf-Zeile(n) die auf allen Seiten oben stehen soll(en), werden nach den Spaltendefinitionen als optionales Argument angenommen:

Spalte 1	Super	Hey
Hallo	Welt	Na
Wie	geht	es
dir	denn?	Ich
mag	Züge	!
Und	Ich	bin
eine	echt	lange
Tabelle	findest	du
nicht	auch	?
Shubi	dubi	duuu

▼	Spalte 1	Super	Hey
	Das ist doch	Wirklich super	tolle dolle
	Mir fehlen	die Worte	Ich hörte
	Die Tabelle	sehr lange	oh bange!

Ich habe den Code hier einmal separiert, um auch wirklich einen Zeilenumbruch an der Stelle zu forcieren können:

```

1 \begin{mtable}{llr}[Spalte 1 & Super & Hey]
2 Hallo & Welt & Na \\
3 Wie & geht & es \\
4 dir & denn? & Ich \\
5 mag & Züge & ! \\
6 Und & Ich & bin \\
7 eine & echt & lange \\
8 Tabelle & findest & du \\
9 nicht & auch & ?
10 \end{mtable}

```

Der zugehörige kleine Pfeil in den fortgeföhrten Kopfzeilen definiert der Befehl `\MNHeaderRow`. In der Regel sollte man annehmen, dass die Verwendung von `env@mtable` einen zweiten Kompiliervorgang erfordert.

◊ `\MHeaderRow`, `\MNHeaderRow`

v2.0.0

Definieren die Gestalt der Kopfzeilen, wobei `\MNHeaderRow` speziell nach Seitenbrüchen angewandt wird. Hier gilt zu beachten, dass die Umgebungen `env@mtable` und `env@mtable` diese nur als Standardeinstellung nutzen und somit auch die Verwendung der Befehle für einzelne Argumente ausgeschaltet werden kann. Standardmäßig unterscheiden sich die beiden Befehle übrigens nur in der Hinsicht, dass `\MNHeaderRow` noch ein kleines Dreieck an die linke Tabellenseite setzt. Sie verwenden die Farben `HeaderColor` sowie `NextHeaderColor`, die initial beide auf `\Hcolor` gesetzt werden, aber jederzeit überschreibbar sind.

Bemerkung 24 – Der versuchte Ersatz ☺

Um den Problemen von `tabu` entgegenzuwirken liefert LILLYxTABLESxMATERIAL noch zwei provisorische Umgebungen. Diese verwenden `\@@tabularArgPatch@iter`, welches automatisch die für `\setrow` benötigten Konfigurationen ^ und + einfügt (allerdings auch vor Befehlen und anderen Zeichen was bedeutet, dass wirklich nur Spalten akzeptiert werden. Dies hat den Nachteil, dass neue Spalten erstellt werden müssen, wenn verschiedene Befehle gewünscht sind!).

◊ `env@mtabular[*]` [`Headerrow=⟨\MTBHeaderRow⟩`] [`PreCode`] {`Header`}

v2.0.0

[`FirstRow=⟨MudWhite!10⟩`] [`Second Row=⟨MudWhite!90⟩`]

Setzt eine Tabelle in einem an das Material-Design des Editor Atom angelehnten Design:

```

1 \begin{mtable}{lll}
2   Hallo & Welt \\
3   Na & Wie \\
4   geht & es \\
5   dir & denn?
6 \end{mtable}

```

Hallo	Welt
Na	Wie
geht	es
dir	denn?

Die Variante mit einem Stern, besitzt keine explizite Formatierung der Kopfzeile. Es gilt zu beachten, dass Lilly für diese Tabellen eine eigene Version des Pakets `xcolor` (<https://www.ctan.org/pkg/xcolor>) mitführt, um auch ältere Latex-Versionen zu unterstützen! Sollte dieses Verhalten zu Problemen führen, so kann dies durch die Definition von `\LILLYxUSERREGXCOLOR` vor dem Einbinden von Lilly deaktiviert werden!

◇ `env@mtable[*][Headerrow=(\MTBHeaderRow)][PreCode]{Header}[Mulitpage Header][FirstRow=(MudWhite!10)][Second Row=(MudWhite!90)][Next Headerrow=(\MNTBHeaderRow)]` v2.0.0

Setzt eine Tabelle in einem an das Material-Design des Editor Atom angelehnten Design:

Spalte 1	Super	Hey
Hallo	Welt	Na
Wie	geht	es
dir	denn?	Ich

Und hier der Code:

```

1 \begin{mtable}{lll}[Spalte 1 & Super & Hey]
2   Hallo & Welt & Na \\
3   Wie & geht & es \\
4   dir & denn? & Ich
5   % ...
6 \end{mtable}

```

Für die Realisierung wird `longtable` verwendet!

◇ `\MTBHeaderRow, \MNTBHeaderRow` v2.0.0

Diese Befehle funktionieren analog zu `\MHeaderRow` und `\MHeaderRow`, wobei sie die Befehle so abändern, dass sie in den Kompatibilitätsumgebungen `env@mtable` und `env@mtable` funktionieren!

8.4 Gedichte

Diese Definitionen werden über die Bibliothek `LILLYxPOEMS` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen. Dieses Modul macht von `LILLYxORNAMENTS` gebrauch!

◊ \poemssetauthor{Author}

v2.0.0

Setzt den Autor, der standardmäßig auf \AUTHOR gesetzt ist und so als Standardautor für alle folgenden Gedichte verwendet wird.

◊ env@poem[Comment][Left Ornament][Right Ornament]

v2.0.0

{Name}{Date}[Author=\poems@author]

Setzt ein Gedicht mit dem Namen Name geschrieben am Date. Der Autor kann durch \poemssetauthor geändert, oder gezielt durch Author angegeben werden. Diese Umgebung unterscheidet sich von env@poems*, dass sie noch einen Seitenumbruch einfügt und das Gedicht auch vertikal zentriert. Allein deswegen wird zum Beibehalt der Dokumentationskonsistenz dort ein entsprechendes Beispiel gezeigt, da sich die Umgebungen sonst gleichen.

◊ env@poem*[Comment][Left Ornament][Right Ornament]

v2.0.0

{Name}{Date}[Author=\poems@author]

Für eine genaue Erklärung siehe env@poem. Hier ein Beispiel:

```
1 \begin{poem*}{Hallo Welt}{10.09.2019}[Florian Sihler]
2 Hallo Welt,
3 Wie geht es dir?
4 Du bist so fern,
5 und doch bei mir.
6 ich hab dich gern,
7 und bin doch hier.
8 \end{poem*}
```

**Hallo Welt**

Hallo Welt,
Wie geht es dir?
Du bist so fern,
und doch bei mir.
ich hab dich gern,
und bin doch hier.



◊ \subduelines

v2.0.0

Wie am Beispiel für env@poem* zu sehen ist, müssen keine Zeilenenden oder ähnliches

angegeben werden. Dies mag die Verwendung von `\obeylines` vermuten lassen, allerdings hätte dieser Befehl den Nachteil, dass die Strophen nicht durch eine Leerzeile getrennt werden könnten, da diese von `\obeylines` geschluckt werden. Deswegen stellt dieser Befehl eine Variante dar, die sowohl in zentrierten als auch nicht zentrierten Umgebungen funktioniert:

```

1 Mit \blankcmd{subduelines}:
2 {
3   \subduelines
4   Hallo Welt,
5
6   Na wie,
7   geht
8
9   es dir?
10 }
11
12 \textbf{Zum Vergleich: }\\
13 Mit \blankcmd{obeylines}:
14
15 {
16   \obeylines
17   Hallo Welt,
18
19   Na wie,
20   geht
21
22   es dir?
23 }
```

Mit `\subduelines`:
 Hallo Welt,
 Na wie,
 geht
 es dir?
Zum Vergleich:
 Mit `\obeylines`:
 Hallo Welt,
 Na wie,
 geht
 es dir?

◊ env@quotes[*Title=*{*Quotes*}][*Author=*{\poems@author}]

v2.0.0

Läutet eine Umgebung ein, in der verschiedene, durch `\singlequote` oder `env@quote` eingeläutete Zitate definiert werden können. Im Gegensatz zu `env@quotes*`, setzt `env@quotes` einen Rahmen um die Zitate:

```

1 \begin{quotes}
2   \begin{quote}
3     Ein Dieter kommt selten zu zweit.
4   \end{quote}
5   \begin{quote}[14.42.2422]
6     Niemand hat die Absicht die Erde zu sprengen!
7   \end{quote}
8 \end{quotes}
```

Ergibt (der Seitenumbruch wird erzwungen ☺):



„Ein Dieter kommt selten zu zweit.“

Florian Sihler

„Niemand hat die Absicht die Erde zu sprengen!“

Florian Sihler – 14.42.2422



◊ `env@quotes*[Author=⟨\poems@author⟩]`

v2.0.0

Definiert eine Umgebung in der der Befehl `\singlequote` und die Umgebung `env@quote`. Diese Umgebung wird von `env@quotes` intern verwendet.

```
1 \begin{quotes*}
2   \singlequote{Hallo}
3   \singlequote{Welt}
4 \end{quotes*}
```

„Hallo“
„Welt“

◊ `\singlequote{quote}`

v2.0.0

Setzt innerhalb einer `env@quotes*`-Umgebung ein einzelnes Zitat, ohne gesonderte Autor oder Datumsangabe, so können mehrere Zitate mit gleichen Metadaten hübsch gesetzt werden:

◊ `env@quote[Date]`

v2.0.0

Setzt unter der Verwendung von `\singlequote` ein Zitat mit Autor und optionales Datumsangabe.

Bemerkung 25 – Auflistung der Gedichte und Zitate

Alle Gedichte können mit `\listofPOEMS`, alle Zitate mit `\listofQUOTES` gelistet werden.

8.5 Stundenpläne

Diese Definitionen werden über die Bibliothek `LILLYxPLILLYxTIMETABLESOEMS` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen.

◊ `\NewTimeTable[tt keys]{Name}`

v2.0.0

Erzeugt einen neuen Stundenplan, dessen Werte mittels `\xdef` persistiert werden (Präfix: `lillyxTIMETABLESx`). Neben dem Zähler `@<Name>@maxcount` werden die beiden Befehle `\the<Name>` und `\present<Name>` erzeugt.

Es gibt eine ganze Reihe möglicher `tt keys` die übergeben werden können und das endgültige Aussehen des Stundenplans konfigurieren.

Bezeichner	Typ	Standard	Beschreibung
title	String	Ich bin ein...	Titel des Stundenplans
bordercolor	Farbe	black	Rahmenfarbe
day start time	Zahl (0-23)	8	Tagesstartzeit
day end time	Zahl (0-23)	20	Tagesendzeit
week start day	Zahl (0-6)	0	Starttag (Montag)
week end day	Zahl (0-6)	4	Endtag (Freitag)
time formatter	enum (siehe unten)	AtLineDistCompanion	Formatierung der Zeit

Der `time formatter` ist in sofern besonders, dass er zwei Makros fordert, die der Syntax `@@TimeFormatter@<Bezeichner>` und `@@TimeFormatter@<Bezeichner>@Sequence` folgen und so definieren wie die Zeiten am linken Rand formatiert werden sollen. Die folgenden Werte sind bereits vordefiniert: `Default`, `RevertDefault`, `AtLine`, `AtLineCompanion` und `AtLineDistCompanion`. Betrachten wir einmal ein Beispiel:

```

1 \NewTimeTable[%  

2   title = Hallo Welt,  

3   day start time = 12, % Start 12 uhr  

4   day end time = 16, % Ende 16 uhr  

5   week start day = 5, % Samstag und  

6   week end day = 6, % Sonntag  

7   time formatter = Default  

8 ]{TimeTable}  

9 \begin{center}  

10   \theTimeTable  

11 \end{center}

```

Hallo Welt

	Samstag	Sonntag
14 - 14		
12 - 12		
10 - 16		
14 - 16		
16		

Ein weiteres Beispiel:

```
1 \NewTimeTable{AnotherTimeTable}
```

```

2 \begin{center}
3   \theAnotherTimeTable
4 \end{center}

```

Ich bin ein Titel, setzte 'title' um mich zu ändern

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

◊ `\the<Name>`, `\present<Name>`

v2.0.0

Rufen jeweils `\DrawTimeTable` beziehungsweise `\PresentTimeTable` mit dem entsprechenden Namen als Argument auf.

◊ `\DrawTimeTable[TimeTable]`

v2.0.0

Zeichnet den entsprechenden Stundenplan innerhalb einer `env@tikzpicture`-Umgebung.

◊ `\PresentTimeTable[TimeTable]`

v2.0.0

Setzt den Stundenplan auf eine neue Seite und setzt die Abstände entsprechend. Intern wird auf `\DrawTimeTable` zurückgegriffen.

◊ `\RawTimeTableEvent[tt event keys]{TimeTable}`

v2.0.0

Generiert ein Event im Timetable, wobei automatisch zwei verschiedene Stile je nach Länge des Events angewendet werden.

Bezeichner	Typ	Standard	Beschreibung
------------	-----	----------	--------------

<code>title</code>	<i>String</i>	bummelbahn	Titel des Events
<code>short title</code>	<i>String</i>		Kurzer Titel des Events
<code>bgcolor</code>	<i>Farbe</i>	AppleGreen!15	Rahmenfarbe
<code>day</code>	<i>Zahl (0-6)</i>	0	Tag des Events
<code>y</code>	<i>Zahl</i>	-1.125	Vertikale Position des Events
<code>height</code>	<i>Zahl</i>	1.25	Länge des Events
<code>preCode</code>	<i>Code</i>		Code vor dem Event
<code>postCode</code>	<i>Code</i>		Code nach dem Event
<code>extra 1</code>	<i>String</i>	Hamsterbacke	Text links unten
<code>extra 2</code>	<i>String</i>	Waffeln	Text rechts unten
<code>extra 3</code>	<i>String</i>	Günther der...	Text mitte

Die Erzeugung eines solchen Events generiert einen neuen Befehl mit dem Bezeichner `\@<TimeTable>@event@id`, wobei die `id` ein hochzählender Wert ist.

Wie leicht ersichtlich ist, ist dieser Befehl zwar öffentlich, aber nicht sehr angenehm direkt benutzt zu werden. Deswegen:

◊ `\NewTimeTableEvent{EventID}{Titel}{Farbe}[Extra 1][Extra 2]` v2.0.0
`[Extra 3][Length=<1.25>]`

Erzeugt den Befehl `\new<EventID>`, wobei dieser an die Konstruktion von `\RawTimeTableEvent`, die definierten Einstellungen als Standart übernimmt:

```

1 \NewTimeTable[title=Hi,week end day=1, day end time=12]{TestTable}
2
3 \NewTimeTableEvent{EinEvent}{Hallo Welt}{bondiBlue!25}
4
5 \newEinEvent{TestTable}{Dienstag}{8 uhr}
6
7 \begin{center}
8   \theTestTable
9 \end{center}
```

Information: Die Notation mit `Dienstag` und `8 uhr` entstammt der standardmäßig ebenfalls eingebundenen Bibliothek `LILLYxTIMETABLESxCOMFORT`, die weiter unten beschrieben wird.

Hi

	Montag	Dienstag
8		Hallo Welt
9		
10		
11		

◊ `\new<EventID>[tt keys]{TimeTable}{Day}{Hour}`

v2.0.0

Registriert ein neues Event mittels `\RawTimeTableEvent` im TimeTable am Day um Hour.

8.5.1 Komfort

Diese Definitionen werden über die Bibliothek `LILLYxTIMETABLESxCOMFORT` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen und basieren auf `LILLYxTIMETABLES`.

Dieses Paket erweitert die Definition von einem Event durch `\NewTimeTable` oder vergleichbaren Befehlen in sofern, dass sie für den Tag die Bezeichner `Monday`, `Tuesday`, ..., `Sunday`, `Montag`, `Dienstag`, ..., `Sonntag` und `Mo`, `Di`, ..., `So` zulassen und für die Zeit die Bezeichner `0 uhr`, `1 uhr`, ..., `23 uhr` akzeptiert, die jeweils in entsprechende Werte für `day` und `y` übersetzt werden.

8.5.2 Universitäts Stundenpläne

Diese Definitionen werden über die Bibliothek `LILLYxTIMETABLESxUNIVERSITY` zur Verfügung gestellt. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxPRESENTER` geladen und basieren auf `LILLYxTIMETABLES`.

◊ `\@@CreateNewLectureEvent{Lecture ID}`

v2.0.0

`{Event Length}{bgcolor}{Moderator}`
`{where}{title}{signature}{name}{short}`

Erstellt einen neuen Befehl der Signatur `\<signature><Lecture ID>`, der die gleichen Argumente wie `\new<EventID>` akzeptiert. Der Befehl wird von `\NewLectureSeries` erstellt.

◊ `\NewLectureSeries[uni keys]{Lecture ID}{Title}{Docent}`

v2.0.0

Definiert eine neue Vorlesungsreihe, die eine ganze Reihe an keys erwartet, auf derer Basis (standardmäßig) ein Befehl für die Vorlesung, Übung und für das Tutorium erstellt wird, sofern nicht anders konfiguriert.

Bezeichner	Typ	Standard	Beschreibung
<code>title</code>	<code>String</code>		Titel der Vorlesungsreihe

<code>short title</code>	<code>String</code>		Kürzel der Vorlesungsreihe
<code>docent</code>	<code>String</code>		Dozent der Vorlesungsreihe
<code>exercise instructor</code>	<code>String</code>	None	Übungsleiter der Vorlesungsreihe
<code>tutor</code>	<code>String</code>	None	Tutor der Vorlesungsreihe
<code>vl length</code>	<code>enum (siehe unten)</code>	2 hours	Länge der Vorlesung
<code>vl bgcolor</code>	<code>color</code>	AppleGreen!25	Farbe der Vorlesung
<code>vl where</code>	<code>String</code>		Ort der Vorlesung
<code>vl title</code>	<code>String</code>	Vorlesung	Bezeichner der Vorlesung
<code>vl signature</code>	<code>String</code>	vl	Präfix des Befehls
<code>vl enabled</code>	<code>Boolean</code>	true	Wenn false, wird kein Befehl erstellt.
<code>ub length</code>	<code>enum (siehe unten)</code>	2 hours	Länge der Übung
<code>ub bgcolor</code>	<code>color</code>	ChromeYellow!15	Farbe der Übung
<code>ub where</code>	<code>String</code>		Ort der Übung
<code>ub title</code>	<code>String</code>	Übung	Bezeichner der Übung
<code>ub signature</code>	<code>String</code>	ub	Präfix des Befehls
<code>ub enabled</code>	<code>Boolean</code>	true	Wenn false, wird kein Befehl erstellt.
<code>tu length</code>	<code>enum (siehe unten)</code>	1 hour	Länge des Tutoriums
<code>tu bgcolor</code>	<code>color</code>	ChromeYellow!15	Farbe des Tutoriums
<code>tu where</code>	<code>String</code>		Ort des Tutoriums
<code>tu title</code>	<code>String</code>	Tutorium	Bezeichner des Tutoriums
<code>tu signature</code>	<code>String</code>	tu	Präfix des Befehls
<code>tu enabled</code>	<code>Boolean</code>	true	Wenn false, wird kein Befehl erstellt.

Ein Beispiel:

```

1 \NewLectureSeries[%  

2   short title=Ana,  

3   vl length = 2 hours, % default  

4   vl where = Raum 42,  

5   exercise instructor = Frau Zensiert,  

6   ub length = 1 hour, %  

7   ub where = Raum 42,
```

```

8   tutor = Herr Zensiert,
9   tu length = 2 hours,
10  tu where = Raum 26
11 ]{anaI}{Analysis für Inf. und Ing.}{Zensiert}
12
13 \NewLectureSeries[%]
14   short title=GdBS,
15   vl length = 2 hours, % default
16   vl where = Raum 123,
17   % We will set the übung to be the labor :d
18   exercise instructor = Dr. Zensiert,
19   ub length = 2 hours,
20   ub where = Ja wo denn?,
21   ub title = Labor,
22   ub signature = lb, % will be lbgdbs not ubgdbs!
23   ub bgcolor = Veronica!25, % different bg color
24   tutor = Der Vergessene,
25   tu length = 1 hour,
26   tu where = Raum 19
27 ]{gdbS}{Grundlagen der Betriebssysteme}{Prof. Dr. Zensiert}
28
29
30 \NewTimeTable[title=Stundenplan SoSe 19]{Stundenplan}
31
32 % ANA
33 \ubanaI{Stundenplan}{Dienstag}{14 uhr} % day, starttime, length >
34   constructed above :D
35 \vlanaI{Stundenplan}{Donnerstag}{12 uhr}
36 \vlanaI{Stundenplan}{Freitag}{8 uhr}
37 \tuanaI{Stundenplan}{Freitag}{10 uhr}
38
39 % GDBS
40 \vlgdbs{Stundenplan}{Montag}{16 uhr}
41 \vlgdbs[extra 2 = Raum 42]{Stundenplan}{Donnerstag}{16 uhr}
42 \lbgdbs{Stundenplan}{Montag}{12 uhr}
43 \theStundenplan

```

Stundenplan SoSe 19

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
8					Analysis für Inf. und Ing. Zensiert
9					Vorlesung RAUM 42
10					Analysis für Inf. und Ing. Herr Zensiert
11					Tutorium RAUM 26
12	Grundlagen der Betriebssysteme Dr. Zensiert			Analysis für Inf. und Ing. Zensiert	
13	Labor JA WO DENN?			Vorlesung RAUM 42	
14		Ana Übung Frau Zensiert RAUM 42			
15					
16	Grundlagen der Betriebssysteme Prof. Dr. Zensiert			Grundlagen der Betriebssysteme Prof. Dr. Zensiert	
17	Vorlesung RAUM 123			Vorlesung RAUM 42	
18					
19					
20					



CONTROLLER

EINE GROSSER HAUFEN ZAHRÄDER

VER 1.0.0

Dieses Definitionen liegen hier:

`\LILLYxPATHxCONTROLLERS = source/Controllers`

Bemerkung 26 – Controller standalone

Die Controller besitzen aus logischen Gründen kein Paket welches sie alle vereint, da sie in der Regel verschiedene Bereiche abstecken und so nicht komplett eingebundenen werden müssen.

9.1 Umgebungen

Diese Definitionen werden über die Bibliothek `LILLYxCONTROLLERxENVIRONMENT` zur Verfügung gestellt.

Dieser Controller lädt erstmal alle für Umgebungen notwendigen Bestandteile sowie die Pakete `enumerate` und `enumitem`. Glückt das Laden des letzteren Pakets, so werden automatisch einige weitere, im Folgenden aufgelesitete Umgebungen zur Verfügung gestellt.

◊ `env@enumeratea[enumargs=<, >]` WAR Veraltet

v1.0.0

War vor der existenz von `env@aufgaben` der entsprechende Ersatz/Platzhalter. Setzt eine in lateinischen Kleinbuchstaben nummerierte Liste:

```
1 \begin{enumeratea}
2   \item Hallo
3   \item Welt
4   \item Na
5   \item du?
6 \end{enumeratea}
```

- a) Hallo
- b) Welt
- c) Na
- d) du?

◊ `env@ditemize[enumargs]`

v1.0.4

Setzt eine unsortierte Auflistung auf Basis von `ditemize`, allerdings sind die Symbole auf den jeweiligen Verschachtelungstiefen, bis zu einer Tiefe von 4 angepasst:

```

1 \begin{ditemize}
2   \item Hallo \begin{ditemize}
3     \item Noch \begin{ditemize}
4       \item Hu
5       \item Hu
6     \end{ditemize}
7     \item besser
8   \end{ditemize}
9   \item Welt
10 \end{ditemize}

```

◊ Hallo
 ♡ Noch
 – Hu
 – Hu
 ♡ besser
 ◊ Welt

◊ env@aufgaben [Spalten] [enumargs]

v2.0.0

Präsentiert eine Liste in Anlehnung an env@enumerate, erlaubt allerdings eine Erweiterung durch env@multicols, weiter wurden Abstände angepasst:

```

1 \begin{aufgaben}
2   \item Wichtig!
3   \item Super $a+b^2$ 
4   \item na du?
5   \item schwer!
6 \end{aufgaben}
7
8 \begin{aufgaben}[2]
9   \item Wichtig!
10  \item Super $a+b^2$ 
11  \item na du?
12  \item schwer!
13 \end{aufgaben}

```

- a) Wichtig!
 - b) Super $a + b^2$
 - c) na du?
 - d) schwer!
- | | |
|--------------------|------------|
| a) Wichtig! | c) na du? |
| b) Super $a + b^2$ | d) schwer! |

Siehe Übungsblatt für ein eingebettetes Beispiel.

9.2 Worttrennung

Diese Definitionen werden über die Bibliothek LIB LILLYxCONTROLLERxHYPHEN zur Verfügung gestellt.

Dieser Controller ist an sich eine datensammlung von Wörtern, für die bisher keine/oder (nach dem Duden) falsche Trennungsvorgaben existieren. Es wird biebei auf die Funktionen des Pakets hyphenat zurück gegriffen. Eine Auflistung aller bisher getrennten Worte findet nicht statt!

9.3 Verlinkungen

Diese Definitionen werden über die Bibliothek **LILLYxCONTROLLERxLINK** zur Verfügung gestellt. Weiter wird mit den anderen Paketen **LILLYxCONTROLLERxMODE** und **LILLYxCONTROLLERxLINK** zusammengearbeitet.

◊ \LILLYxHYPERLINK

v1.0.2

Ist dieser Befehl auf **\true** gesetzt, werden die Links in herkömmlicher Variante gesetzt (farbig, klickbarer Link). Wird der Wert auf **\false** gesetzt, so werden die Links in der Druckmanier durch die Angabe der Seitennummer gesetzt.

◊ \setLinkColor{Color}, \lpage{Page Number}

v2.0.0

Ersterer Befehl setzt die aktuelle Farbe für einen Link, es gilt zu beachten, dass **\jmark**, **\hmark** und **\cmark** diesen Wert jeweils überschreiben. Kann in Kombination mit **\lpage** verwendet werden, welches auf die Seite mit der entsprechenden Nummer verweist. Ein Beispiel:

```
1 Seite \setLinkColor{bondiBlue}\lpage{5} % → Seite 5
```

◊ \elable[*]{Name}, \elabel[*]{Name}

v1.0.0

Setzt einen Anker für Linkverknüpfungen, wobei dieser automatisch in von **LILLYxBOXES** definierten Boxen auf den Start der Box gesetzt wird. Ist dies nicht gewünscht (soll also wirklich genau zum gesetzten Befehl gesprungen werden), genügt das Platzieren des Sterns. Intern wird **\label** verwendet.

◊ \jmark{Name}{Ziel}, \hmark{Name}{Ziel}, \silentHmark{Name}{Ziel}[color]

v1.0.0

Erzeugen Sprungmarken zu einem Ziel:

```
1 \elable{Hallo}
2
3 Hey: \jmark[Hallo Welt]{Hallo} \\
4 Ho: \hmark[Hallo Welt]{Hallo} \\
5 Jeah: \silentHmark[Hallo Welt]{Hallo}[Ao]
```

Hey: Hallo Welt
Ho: Hallo Welt
Jeah: Hallo Welt

Übrigens, hier das Ergebnis der Druckversion (\LILLYxHYPERLINK auf \false):

Hey: Hallo Welt → 101

Ho: **Hal**lo Welt → 101

Jeah: Hallo Welt

◊ \cmark{Name}{Ziel}{Farbe}

v2.0.0

Setzt den Link wie **\jmark**, allerdings ohne **\LILLYxHYPERLINK** zu beachten.

◊ \eXButton[Command]{Name}

v1.0.2

Setzt einen Hyperlink mit Funktionen, die eigentlich dem PDF-Viewer vorbehalten sind. Deswegen hängt die Unterstützung auch vom verwendeten Viewer ab. Beispiel: **\eXButton{Find}{\faSearch}** ergibt: 

9.4 Modi-Kontrolle

Diese Definitionen werden über die Bibliothek **LILLYxCONTROLLERxMODE** zur Verfügung gestellt.

- ◊ **\LILLYxMODE**, **\LILLYxMODExDEFAULT**, **\LILLYxMODExPRINT**, **\LILLYxMODExDUMMY** v1.0.0
Über das Makro **\LILLYxMODE** wird gesteuert, welcher modus verwendet werden soll. Hierbei speichern die anderen Makros **\LILLYxMODExDEFAULT**, ... welchen der jeweiligen Werte **\LILLYxMODE** halten muss: **\LILLYxMODExDEFAULT** (default), **\LILLYxMODExPRINT** (print) und **\LILLYxMODExDUMMY** (dummy).
- ◊ **\LILLYxFOOTERxBUTTONS** v1.0.0
Wird, sobald definiert, auf **\true** gesetzt und kann in manchen Layouts dafür sorgen, dass die PDF-Typischen Buttons (wie in dieser Dokumentation rechts unten) angezeigt werden. Wird, durch das Setzen von **\LILLYxMODE** auf **\LILLYxMODExPRINT** automatisch deaktiviert (auf **\false**) gesetzt.
- ◊ **\LILLYxIMAGESxShow** v1.0.3
Wird auf **\true** gesetzt, wenn **\LILLYxMODExEXTRA** auf **\true** steht. Kann und wird an manchen Stellen verwendet um Grafiken, gezielt entfernen zu können.
- ◊ **\LILLY@Typ@Mitschrieb**, **\LILLY@Typ@Uebungsblatt**, **\LILLY@Typ@Dokumentation**, **\LILLY@Typ@Zusammenfassung** [WAR Veraltet] v1.0.0
Definieren bis **VER 1.0.9** die Werte, die **\LILLY@Typ** halten muss, um den jeweiligen Modus zu Laden: **\LILLY@Typ@Mitschrieb** (MITSCHRIEB), **\LILLY@Typ@Uebungsblatt** (UEBUNGSBLATT), **\LILLY@Typ@Dokumentation** (DOKUMENTATION) und **\LILLY@Typ@Zusammenfassung** (ZUSAMMENFASSUNG).

9.5 Layout Kontrolle

Diese Definitionen werden über die Bibliothek **LILLYxCONTROLLERxLAYOUT** zur Verfügung gestellt. Der Layout-Controller übernimmt mit **VER 2.0.0** die Aufgaben der **LILLYxCONTROLLERxINTRO** und der **LILLYxCONTROLLERxOUTRO** Pakete, die noch für diese Version mit dem Präfix **DEPRECATED_** mitausgeliefert werden.

Bemerkung 27 – Layouts

Ein Layout, welches von der Layout-Verwaltung akzeptiert werden möchte, benötigt den Namen **_LILLY_LAYOUT_<Bezeichner>**, wobei **<Bezeichner>** ein frei wählbarer Name ist, unter dem sich das Layout von da an ansprechen lässt. Gesucht wird (mittels **\userput**) in den Pfaden **\lillyPathLayout** und **\LILLYxPATHxDATA/Layouts**, wobei der letzte die von Lilly mitgelieferten Layouts enthält, die weiter unten vorgestellt werden.

Ist **\LILLYxDEBUG** auf **\true** gesetzt, wird **\errorcontextlines** entsprechend modifiziert und eine entsprechende DEBUG-Titlepage gesetzt:

DEBUG-TITLEPAGE

Allgemein:

Lilly: 2.0.0 – Jake ist auch nur Java	\LILLYxVERSIONxLONG
Datum: 20. September 2019	\heute
Kompilier-Modus: default	\LILLYxMODE
Box-Modus: LIMERENCE	\LILLYxBOXxMODE
Farbprofil: 	\LILLYxCOLORxRainbow

PDF-Spezifisch:

Author: Florian Sihler	\AUTHOR
Email: florian.sihler@web.de	\AUTHORMAIL
Dokumentname: Lilly-Dokumentation.doc.tex	\LILLYxDOCUMENTNAME
PDF-Name: ./Lilly-Dokumentation.doc	\LILLYxPDFNAME
Output: ./	\LILLYxOUTPUTDIR
Input: ./	\LILLYxPATH
External: FALSE	\LILLYxEXTERNALIZE
Bibtex ⁽¹⁾ :	\LILLYxBIBTEX

Jake:

ThreadID ⁽²⁾ :	\LILLYxTHREADxID
Semester: 0	\LILLYxSemester
Vorlesung: GDRA	\LILLYxVorlesung
Highlight-Color: DebianRed!85	\Hcolor
Layout Path: source/Data/Layouts	\lillyPathLayout

⁽¹⁾No Value means bibtex was disabled.

⁽²⁾No Value means compilation wasn't parallelized.

Ist \LILLYxSemester auf einen Wert größer als 0 gesetzt, wird automatisch die Konfiguration mittels von \RequestConfig geladen. Ist weiter \LILLYxBIBTEX definiert, so werden alle für die Verwendung von BIBTEX notwendigen Pakete (namentlich cite) geladen und automatisch ein Aufruf von bibtex initiiert.

- ◊ \LILLYxCLEARxHEADFOOT v2.0.0
Arbeitet analog zu \clearscrheadfoot, löscht die einzelnen Komponenten allerdings explizit.
- ◊ \heute v2.0.0
Liefert das aktuelle Datum in hübscher deutscher Notation. Dieser Befehl wird gegebenenfalls in ein anderes Paket übertragen: \heute liefert: 20. September 2019.

◊ `\printbib{Name}`

v2.0.0

Setzt die Bibliographie mit dem Namen `Name` automatisch, greift intern auf `\bibliography` zurück, was es auch problemlos ermöglicht diesen Befehl direkt zu verwenden.

9.5.1 Das Mitschrieb-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_MITSCHRIEB`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxCONTROLLERxLAYOUT` auf Basis von `\LILLYxMODE` präsentiert.

Dieses Design ist als Urdesign zusammen mit `LILLYxPHILOSOPHER` für die Generierung von Mitschreiben verantwortlich. Im Folgenden wird nicht auf jede einzelne Modifikation sondern nur auf die nutzbaren Befehle eingegangen:

Inhaltsverzeichnis	
1 Mathe	1
2 Überschrift und Ebene 0 (chapter)	3
2.1 Überschrift und Ebene 1 (section)	3
2.1.1 Überschrift und Ebene 2 (subsection)	4
2.2 Lässt ...	5
2.2.1 Beispiel einer Lässt (normal)	7
2.2.2 Beispiel einer Lässt (innermost)	10
2.2.3 Beispiel einer Lässt (descriptive)	11

Florian Sihler 17. September 2019

Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

$$\pi_{12} = \frac{4 - \sqrt{17 - 4m}}{2n} = \frac{4 - \sqrt{17 - 4n}}{2}$$

Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

$$\sum_{n=1}^{\infty} a_n q^n = \lim_{N \rightarrow \infty} \sum_{n=1}^{N+1} a_n q^n = \lim_{N \rightarrow \infty} a_1 \frac{1-q^{N+1}}{1-q} = \frac{a_1}{1-q}$$

Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Mathe ◀ ▶ ⌂ 1/10

Florian Sihler 17. September 2019

Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

$$\int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi} \int_{-\infty}^{+\infty} e^{-x^2} dx = \int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi}$$

Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Mathe ◀ ▶ ⌂ 2/10

Florian Sihler 17. September 2019

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 2 (subsection) ◀ ▶ ⌂ 1/10

Florian Sihler 17. September 2019

Das hier ist der zweite Absatz. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 2 (subsection) ◀ ▶ ⌂ 2/10

Florian Sihler 17. September 2019

Das hier ist der zweite Absatz. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 3 (chapter) ◀ ▶ ⌂ 1/10

Florian Sihler 17. September 2019

Das hier ist der zweite Absatz. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 3 (chapter) ◀ ▶ ⌂ 2/10

Florian Sihler 17. September 2019

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 0 (chapter) ◀ ▶ ⌂ 4/10

Florian Sihler 17. September 2019

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Das hier ist ein Blindsight zum Testen von Textangaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert des Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindsight“ oder „Handout gefüllt“? Kijk – minutiösen! Ein Blindsight bietet nur wichtige Informationen. An ihm mense ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindsight sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gestellt sein. Es muss keinen Sinn ergaben, sollte aber lesbar sein. Fründsprechige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kapitel: Überschrift auf Ebene 0 (chapter) ◀ ▶ ⌂ 6/10

◊ `\TitleSUB{Text}`

v2.0.0

Erlaubt es einen Text als Titelunterschrift zu setzen. Siehe [hier](#) (Beginn dieses Kapitels) für ein Beispiel.

Kapitel Controller

Q C C ◀ 104/149 ►

Bemerkung 28 – End-Hooks

Dieses Layout fügt am Ende automatisch Auflistungen aller Definitionen, Sätze, Lemmata, Zusammenfassungen und Übungsblätter, sofern diese im Dokument auftauchen. Siehe hierfür **LILLYxBOXES**.

Da mit **VER 2.0.0** der Stil für Sektionen (und darunterliegende Level) angepasst wurde, kann man durch das Setzen von `\iflilly@mitschrieb@sectionlines@useold@` auf *true*, den alten Stil zurück erhalten. Hierzu genügt zu Beginn des Dokuments:

```
1 \makeatletter
2   \lilly@mitschrieb@sectionlines@useold@true
3 \makeatother
```

9.5.2 Das Übungsblatt-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_UEBUNGSBLATT`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxCONTROLLERxLAYOUT** auf Basis von `\LILLYxMODE` präsentiert.

Bemerkung 29 – Ein Übungsblatt mit *Jakc*

Wie in der Einleitung bereits angezeigt, ist ein Übungsblatt, wenn man es mit *Jakc* verwendet, der reine TeX-Code. In diesem Falle bettet *Jakc* die Datei nämlich in eine andere gemäß der folgenden Struktur ein:

```
1 \documentclass[Uebungsblatt,Vorlesung=${VORLESUNG},n=${N},Semester,
= ${SEMESTER}]{Lilly}
2 \begin{document}
3   \input{${INPUTDIR}(${TEXFILE})}% Das Übungsblatt
4 \end{document}
```

Das bedeutet natürlich, dass man sich auch selbst ein Übungsblatt ohne *Jakc* basteln kann.

◊ \TUTORBOX

v1.0.1

Setzt eine Tutorbox für Übungsblatt(abgaben) die auf Papier stattfinden:

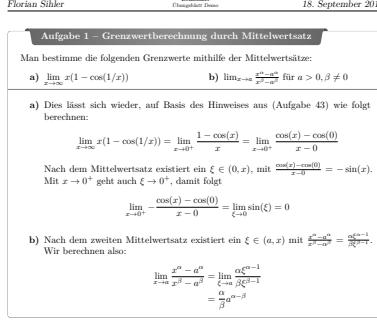
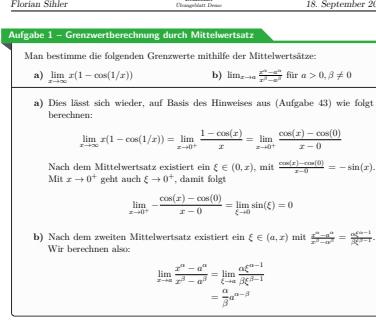
Tutor: Der Unbekannte

◊ \points{Punkte}

v1.0.6

Setzt Punkte an das Ende der Zeile, um zum Beispiel Teilpunkte in Aufgaben einfach zu setzen.

Weiter besteht eine Unterstützung durch **LILLYxBOXES**, so kann der Boxmodi ALTERNATE ein anderers Design hervorrufen (genau genommen durch die Modifikation von `env@aufgabe`):

DEFAULT	ALTERNATE
	
1 / 1	1 / 1

Die Farbe des ALTERNATE-Designs wird durch `\Hcolor` und damit durch die `\Jaket`-Einstellung `lilly-signatur-farbe` kontrolliert. Beide hier gezeigten Dokumente wurden übrigens nur mit `pdflatex` kompiliert! Die Dokumente finden sich zur Ausführlichkeit im Quellordner der Dokumentation unter `Data/Documents/LayoutUebungsblatt`, wobei sie sich derart ähneln, dass hier exemplarisch das Übungsblatt in der DEFAULT-Variante (gekürzt) aufgeführt ist:

```

1 \def\LILLYxBOXxMODE{DEFAULT}
2 \documentclass[Uebungsblatt]{Lilly}
3
4 \def\UEBUNGSHEADER{\textbf{Demoblatt} \Übungsblatt Demo}
5
6 \begin{document}
7 \begin{aufgabe}{Grenzwertberechnung durch Mittelwertsatz}{5}
8   Man bestimme die folgenden Grenzwerte %...
9   \begin{aufgaben}[2]
10     \item \(\lim_{x \rightarrow \infty} x(1 - \cos(1/x))\)
11     \item \(\lim_{x \rightarrow a} \frac{x^\alpha - a^\alpha}{x^\beta - a^\beta}\) für \(a > 0, \beta \neq 0\)
12   \end{aufgaben}
13 \vSplitter

```

```

14 \begin{aufgaben}
15   \item Dies lässt sich wieder, ...
16   \item Nach dem ...
17 \end{aufgaben}
18 \end{aufgabe}
19 \end{document}

```

9.5.3 Das Zusammenfassungs-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_ZUSAMMENFASSUNG`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von `\LILLYxCONTROLLERxLAYOUT` auf Basis von `\LILLYxMODE` präsentiert.

Das schreiben einer Zusammenfassung unterscheidet sich in gewissem Maßem vom schreiben anderer Dokumente. Aus historischen Gründen wird von einer Verwendung der herömmlichen strukturbefehle wie `\section` abgesehen. An ihre Stelle tritt der folgende Befehl, der sich automatisch im Anhang anpasst:

◊ `\TOP[mark]{Title}{Comment}`

v1.0.3

Setzt im *Hauptteil* einen normalen Start eines neuen Themenbereichs und im *Anhang* einen (im Inhaltsverzeichnis anders aufgeführten) mit Anhang angeführten neuen Bereich. Der Wechsel zwischen Hauptteil und Anhang erfolgt einmalig durch `\startAppendix`.

◊ `\startAppendix`

v1.0.2

Eröffnet den Anhang, alle nun durch `\TOP` angeführten Themen werden nicht in das Inhaltsverzeichnis der Titelseite aufgenommen und im table of contents in die Kategorie Anhang sortiert.

◊ `\kw{Main}, \sw{Main}{Sub}, \sr{Main}{Sub}{Lowest}`

v1.0.2

Fügt den entsprechenden Begriff dem Index mittels `\index` hinzu, gibt ihn allerdings auch direkt aus. Weiter formatiert `\kw` den entsprechenden Begriff Fett. Die mit `\sw` und `\sr` weitergegebenen Gruppierungsbegriffe werden natürlich nicht ausgegeben. Das System an sich ist noch nicht wirklich ausgereift und benötigt hin und wieder ein manuelles Eingreifen.

◊ `\imp, \<, \>, \reg{Text}, \customex{Text}`

v1.0.0

Dies sind einige Kurzbefehle, die ich im Rahmen von Zusammenfassungen oft benötigt habe, sonst allerdings nicht ☺:

```

1 \imp, \<, \>, \mto % ergibt: <, <, >, →
2 \reg{Hallo Welt} % ergibt: Hallo Welt
3 \customex{Hallo Welt} % ergibt: Hallo Welt

```

Der Befehl `\reg` wurde hierbei gezielt für Register ins leben gerufen und setzt den Text in `\LILLYxlistTypeWriter`.

◊ `\negaskip, \negbskip, \TOPskip`

v1.0.9

Setzt abstände entsprechend `\skip` und `\bskip`, es handelt sich hierbei um negativ Abstände, die dann verwendet werden können, wenn mehrere Umgebungen Abstände einführen.

- ◊ `\infot[Text]` v2.0.0
Setzt einen Informationstext in kleiner Schrift. Ich verwende es für gewöhnlich als Kommentar, oder wenn die Informationen nicht in den Anhang passen aber dennoch erwähnt werden sollten.
- ◊ `\aLink` v2.0.0
`{Target}` Das Ziel lässt sich genauso mit `\elable` setzen, allerdings wird das Ziel durch das typische Buch/die Seitenzahl um auf einen Verweis im Anhang hinzuweisen, gesetzt.
- ◊ `env@smalldesc, env@smalldite` v2.0.0
Setzt Varianten von `env@description` und `env@ditemize` in einem kompakteren Format.
- ◊ `\showcase[color][tikz cmds]{Name}{Description}[Bonusnote][Bottomtag]` v2.0.0
Präsentiert eine Information in einem an Karten anmutenden Format.

Bemerkung 30 – Eine beispielhafte Zusammenfassung

Im Folgenden ist nun das Ziel eine eigene Zusammenfassung zu erstellen. In diesem Beispiel muss `\LaTeX` nicht verwendet werden, ein Kompilieren mit `-shell-escape` (beziehungsweise je nach System `-enable-write18`) genügt völlig, damit die automatische Index-unsterstützung von Lilly greifen kann.

```

1 \documentclass[Zusammenfassung, Vorlesung=ANA1, Semester=2]{Lilly}
2
3 \begin{document}
4
5 \TOP{Ich bin ein Titel}{Und das ist Lustig}
6 Wenn hier direkt Text kommt, sowas wie zum Beispiel ein wichtiger >
    \kw{Begriff}, muss nnichts weiter gemacht werden!
7 Auch Definitionen sind kein Problem:
8 \begin{definition}[Wichtig]
9     Dies ist die wichtigste Definition die du je gelesen haben >
        wirst \aLink{mrk:Wichtig}.
10 \end{definition}
11 Eine \T{description}:
12 \begin{smalldesc}
13     \item[Alphabet] Malphaset
14     \item[Betabet] Wetaled
15 \end{smalldesc}
16 Oder auch eine \T{ditemize}-Umgebung:
17 \begin{smalldite}
18     \item Punkt 1
19     \item Punkt 2
20     \item Punkt 7, tihih
21 \end{smalldite}
22 \TOP{Ich bin ein besserer Titel}{Und das ist wirklich Lustig}>
    \TOPskip

```

```

23 \begin{definition}[Heyho]
24   Kommt eine Definition direkt nach einem neuen Thema, so >
      bevorzuge ich das verringern des Abstandes!
25 \end{definition}
26 \customex{Dies war aber nun auch wirklich eine tolle Erklärung, >
      finden sie nicht auch Mister Mister?}
27
28 \startAppendix
29
30 \TOP{Anhang, jippieh ajeah}{Wuuup wuup}
31 \elable{mrk:Wichtig}Im Anhang jetzt noch ein tolles \emph{showcase},
32   :
33 \begin{center}
34   \showcase{Tolle Information}{%
35     Ist das nicht wundervoll? Ich finde es eine tolle Box!
36   }
37   Ich bin auch ein wichtiger \sw[Begriff]{Super Begriff}, und der >
      hier ist ein \sr[Begriff]{Super Begriff}{Noch besserer Begriff},
      ist das nicht \kw[wundervoll]?
38
39 \begin{definition}[Anhangsdefinition]
40 Hallo Günthääär.
41 \end{definition}
42 \end{document}

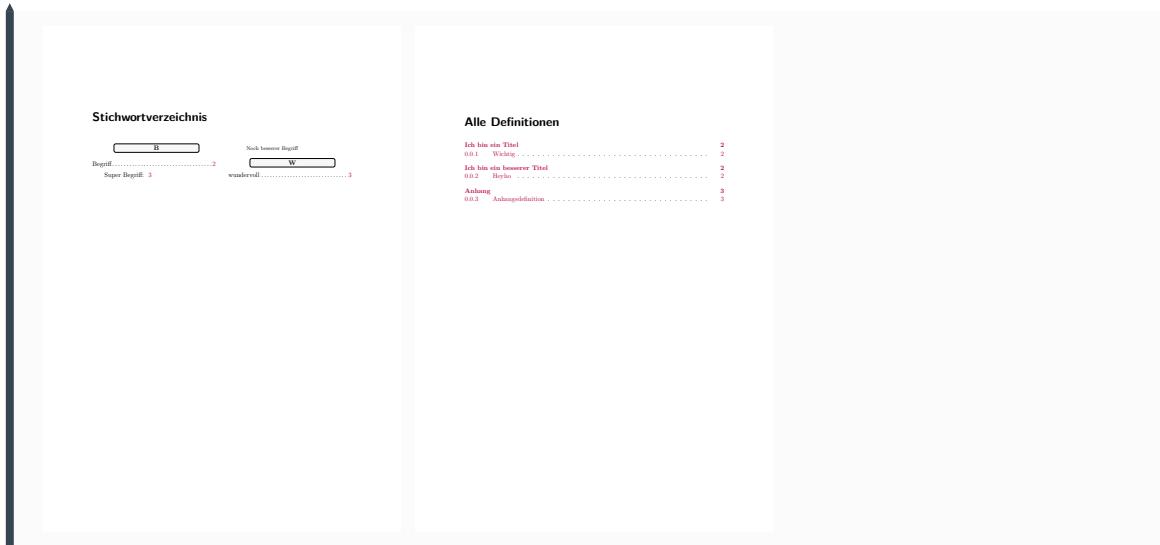
```

Die Generierung der Titelseite erfolgt übrigens ebenfalls mit **LILLYxPHILOSOPHER**. Das (hierbei durch `pdflatex -shell-escape example-zsf.tex` erzeugte) Ergebnis sieht so aus:

The screenshot displays the generated LaTeX document. It includes:

- Zusammenfassung**: A green header bar.
- Analysis 1 für Inf. und Ing.**: The main title, with a subtitle "Dr. Jan-Willem Lübeck".
- Themenübersicht**: Table of contents with the following entries:

	105
Ich bin ein Titel	2
Ich bin ein Absatz	2
Absatz	3
Absatzverzeichnis	4
- Definition 0.1 – Heyho**: Content from line 23 of the code.
- Definition 0.2 – Heyho**: Content from line 37 of the code.
- Definition 0.3 – Anhangsdefinition**: Content from line 41 of the code.
- Footer**: Includes the logo "Mathematik für Informatiker" and the URL "http://github.com/Tagessieger/LILLY".



9.5.4 Das Plain-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_PLAIN`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxCONTROLLERxLAYOUT` auf Basis von `\LILLYxMODE` präsentiert.

Dieses Design wird von Lilly dann gewählt, wenn kein anderes Layout gewählt wird. Es liefert keine Befehle die genutzt werden sollten, sie dienen alle nur der internen Verarbeitung. Hier ein Beispiel:

```

1 \documentclass{Lilly}
2
3 % Aus Test gründen
4 \usepackage{blindtext}
5
6 \begin{document}
7   \blinddocument
8 \end{document}
```

Liefert:

ÜBERSCHRIFT AUF EBENE 0 (CHAPTER)

1

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Gravurtext der Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindtext“ oder „Blauwert gefüllt!“ Kijk! – mitarbeiter! Ein Blindtext bietet mir wichtige Informationen. An ihm mache ich die Lesbarkeit einer Schrift, ihre Ausmusterung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Ausmusterung vermitteln.

1.1 Überschrift auf Ebene 1 (section)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Gravurtext der Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindtext“ oder „Blauwert gefüllt!“ Kijk! – mitarbeiter! Ein Blindtext bietet mir wichtige Informationen. An ihm mache ich die Lesbarkeit einer Schrift, ihre Ausmusterung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Ausmusterung vermitteln.

1.1.1 Überschrift auf Ebene 2 (subsection)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Gravurtext der Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindtext“ oder „Blauwert gefüllt!“ Kijk! – mitarbeiter! Ein Blindtext bietet mir wichtige Informationen. An ihm mache ich die Lesbarkeit einer Schrift, ihre Ausmusterung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Ausmusterung vermitteln.

Überschrift auf Ebene 3 (subsubsection)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst

Fremdsprachige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Ausmusterung vermitteln.

Überschrift auf Ebene 4 (paragraph)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Gravurtext der Schrift an. Ist das wirklich so? Es ist gleichgültig, ob ich schreibe: „Das ist ein Blindtext“ oder „Blauwert gefüllt!“ Kijk! – mitarbeiter! Ein Blindtext bietet mir wichtige Informationen. An ihm mache ich die Lesbarkeit einer Schrift, ihre Ausmusterung, wie harmonisch die Figuren zusammen stehen und präfe, wie breit oder schmal sie lauft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Loren ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Ausmusterung vermitteln.

1.2 Listen

1.2.1 Beispiel einer Liste (bullet)

- Erster Listenpunkt, Stufe 1
- Zweiter Listenpunkt, Stufe 1
- Dritter Listenpunkt, Stufe 1
- Vierter Listenpunkt, Stufe 1
- Fünfter Listenpunkt, Stufe 1

1.2.2 Beispiel einer Liste (*list-item)

- Erster Listenpunkt, Stufe 1
 - Erster Listenpunkt, Stufe 2
 - Erster Listenpunkt, Stufe 3
 - Erster Listenpunkt, Stufe 4
 - Zweiter Listenpunkt, Stufe 4

1.2.3 Beispiel einer Liste (enumerate)

1. Erster Listenpunkt, Stufe 1

2. Zweiter Listenpunkt, Stufe 1

3. Dritter Listenpunkt, Stufe 1

4. Vierter Listenpunkt, Stufe 1

5. Fünfter Listenpunkt, Stufe 1

Beispiel einer Liste (*list-item)

- 1. Erster Listenpunkt, Stufe 1
 - a) Erster Listenpunkt, Stufe 2
 - A. Erster Listenpunkt, Stufe 3
 - B. Zweiter Listenpunkt, Stufe 4
 - c. Dritter Listenpunkt, Stufe 3
 - b) Zweiter Listenpunkt, Stufe 2

1.2.4 Beispiel einer Liste (*description)

Erster Listenpunkt, Stufe 1

Zweiter Listenpunkt, Stufe 2

Erster Listenpunkt, Stufe 2

Zweiter Listenpunkt, Stufe 3

Erster Listenpunkt, Stufe 2

Zweiter Listenpunkt, Stufe 1

9.5.5 Das ElegantBook-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_ELEGANT_BOOK`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LIB LILLYxCONTROLLERxLAYOUT` auf Basis von `\LILLYxMODE` präsentiert.

◊ \TableOfContents

v2.0.0

Setzt den `\tableofcontents` für das ELEGANT_BOOK.

◊ \SetPartFlavour{Text}{Author}

v2.0.0

Setzt die Texte für den nächsten `\part`.

◊ \printMiniToc[Content]

v2.0.0

Setzt einen kleinen `\tableofcontents` für jedes Kapitel, wobei Content, ein beliebiger Inhalt sein kann der ebenfalls auf der Kapitelseite abgebildet wird.

Hier ein Beispiel:

```
1 \documentclass[ElegantBook]{Lilly}
2
3 % Control the main color:
4 \def\Hcolor{DebianRed}
5
6 % Aus Test gründen
7 \usepackage{lipsum}
8
9 \begin{document}
10 % Titlepage oder so
11
12 \TableOfContents
13
14 \SetPartFlavour{\lipsum[2]}{Detlef Dieter}
15 \part{Tolle Welt}
```

```

16
17
18 \chapter{Da wo die Sonne scheint}
19   \printMiniToc
20
21
22 \section{SectionA}
23 % ...

```

Liefert (das volle Beispiel findet sich unter `Data/Documents/LayoutElegantBook/_example-eb.tex`, in den Quelldateien dieser Dokumentation):

The screenshot shows the LaTeX code in a code editor and the resulting PDF output. The PDF is a book layout with two columns per page. The left column contains the table of contents and the right column contains the main content.

Inhaltsübersicht (Table of Contents):

1	1. Tolle Welt
3	3 Da wo die Sonne scheint
7	7 Da wo der Regen weint
11	11 Tolle Gold
13	13 Da wo der Wuffd gränt

Kapitel 1: Da wo die Sonne scheint

Übersicht:

1.1	SectionA	4
1.1.1	SubSectionA	4
1.2	SectionB	5
1.2.1	SubSectionB	5
1.3	SectionC	4
1.3.1	SubSectionC	4

Kapitel 2: Da wo der Regen weint

Übersicht:

2.1	SectionA	8
2.2	SectionB	8
2.2.1	SubSectionB	8
2.3	SectionC	9
2.3.1	SubSectionC	9
2.3.2	SectionD	9

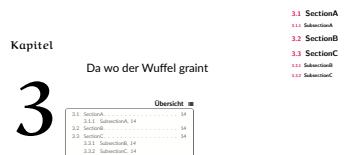
Kapitel 3: Tolles Geld

Übersicht:

3.1	SectionA	8
3.2	SectionB	8
3.3	SectionC	8

Page Headers:

- Page 4: Da wo die Sonne scheint
- Page 5: SectionC
- Page 6: Da wo die Sonne scheint
- Page 7: SectionC
- Page 8: Da wo der Regen weint
- Page 9: SectionC
- Page 10: Da wo der Regen weint
- Page 11: SectionC
- Page 12: Tolles Geld
- Page 13: SectionC
- Page 14: Tolles Geld



Da wo der Wuffel graint

9.5.6 Das Paper-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_PAPER`. Sie werden mit `VER 2.0.0` automatisch mit dem Einbinden von `LILLYxCONTROLLERxLAYOUT` auf Basis von `\LILLYxMODE` präsentiert.

- ◊ `\ABSTRACT, \TITLE, \BRIEF` v1.0.8
Enthalten die jeweiligen Informationen für das Abstract, den Titel und die Kurzbeschreibung des Papers.
- ◊ `\printHeader` v1.0.8
Setzt den Titel des Papers, sollte wohl ganz am Anfang des Dokuments stehen, muss es aber nicht.
- ◊ `\printLILLY` v1.0.9
Setzt einen Disclaimer, dass dieses Dokument mit Lilly generiert wurde. Es ist nicht notwendig dies in das Paper zu setzen, es kann allerdings verwendet werden.
- ◊ `\startAppendix` v1.0.8
Startet den Anhang des Papers.
- ◊ `\intro{Text}` v1.0.8
Setzt einen Text in kuriver Schrift um zum Beispiel eine Kurzzusammenfassung für einen Abschnitt zu gebem.

Im Folgenden ein Beispiel, welches sich auch hier (`Data/Documents/LayoutPaper/example-paper.tex`) in den Quelldateien der Dokumentation findet. Es erzeugt das folgende Dokument:

Setzt eine kleine Erklärung zu gewissen Sektionen oder Abschnitten.
Hier ein Beispiel, der dem Code in Data/Documents/LayoutPnP/example-pnp.tex (der Quelldateien) entspringt:

Tales of a never ending night

Gottaktionen

Kurzübersicht	
1	Gottaktionen für eigene Gläubige
2	Gottaktionen mit der Welt

Abstract.
Die folgenden Seiten liefern eine Übersicht über alle für Punkte erwerbbaren Aktionen. Die hierfür benötigten Punkte werden vom Spielleiter vergeben und verfallen nicht, sofern sie gespart werden. Die Käufe gelten hierbei, sofern nicht anders vermerkt für nur genau eine Provinz, Gottheit, Tierart, ... Im Zweifelsfall empfiehlt es sich die Ideen im Voraus mit dem Spielleiter abzuklären und gegebenenfalls anzupassen.

1 Gottaktionen für eigene Gläubige

- **Ketzerei erklären (Gebot)** 1p
Was wären Religionen ohne Regeln? Genau, LANGWEILIG! Verbiete deinen wahren Gläubigen Dinge und wer sich nicht daranhält, kann ausgestoßen werden. Gesellschaften sind schon super...
- **Orthodoxie erklären (Gebot)** 1p
Was wären Religionen ohne Vorschriften? Genau, LANGWEILIG! Also treibe deine wahren Gläubigen dazu, zu tun was DU WILLST. Wen interessiert es, dass Pilze auf dem Kopf ungesund sind und wahrscheinlich Nährstoffe aus seinem Gehirn saugen? NIEMAND!! DENN GOTT WILL ES!
- Befehle deinen Untertanen, was sie zu tun haben und wann. (Gebot)
- **Wunder vollbringen** 2p
Zeige deinen Anhängern, dass sie dir etwas Wert sind. Hier eine Heilung da ein Ausrotten einer Krankheit. Dort 5000 Brote und ein paar Fische mehr im Wasser.

Wie dieses Wunder aussieht entscheidest du, bespreche es aber bitte mit dem Spielleiter.

2 Gottaktionen mit der Welt

- **Naturwunder erschaffen** 2p
„Uü, toooll“ - Scar, König der Löwen
Es schafft eine Stelle, an der die Welt einfach schöner ist wie dies aussieht bestimmt du selbst (beachte, dass sie zum Biom passen sollte)

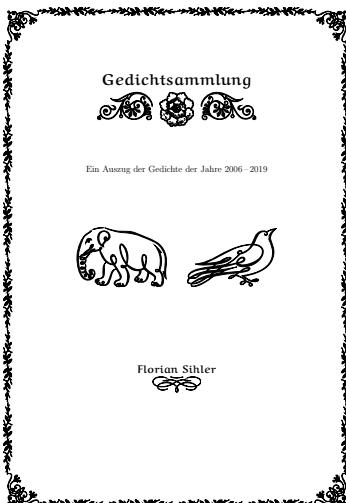
Es wurden für dieses Beispiel noch weitere Umgebungen in der Datei generiert (die Ansicht hier ist beschnitten).

9.5.8 Das Poems-Layout

Diese Definitionen befinden sich in der Datei: `\LILLYxPATHxDATA/Layouts/_LILLY_LAYOUT_POEMS`. Sie werden mit **VER 2.0.0** automatisch mit dem Einbinden von **LILLYxCONTROLLERxLAYOUT** auf Basis von **\LILLYxMODE** präsentiert. Das Layout eignet sich explizit dazu, zusammen mit **LILLYxPOEMS** verwendet zu werden.

- `\poemstoc, \poemstitle[Left Ornament=<ornaelephant>]` [Right Ornament=<ornagoat>]
v2.0.0
Setzt, ähnlich zu `\pnptoc` ein Inhaltsverzeichnis, allerdings unter Unterstützung von **LILLYxORNAMENTS**. Der Titel nimmt hier eine komplette Seite ein und definiert zwei Ornamente (Left Ornament und Right Ornament) die frei definiert werden können.
- `\poemssettitle{title}, \poemssetsubtitle{subtitle},` **v2.0.0**
`\poemssettocpoemsheader{header}, \poemssettocquotesheader{header}`
Setzt die jeweiligen Felder für die Daten.

Im Folgenden ein Beispiel, welches sich wiedereinmal (Data/Documents/LayoutPoems/poems.tex) in den Quelldateien der Dokumentation findet. Es erzeugt das folgende Dokument:



Inhaltsverzeichnis	
1 Hölle Welt, 2	2
2 Sie, 6	Wochensäulen, 14
3 Feind, 7	Die Brüder, 16
4 Freude, 8	Finsternis, 17
5 Schattenküste, 9	Die Seele, 18
6 Schattenkönig, 10	Winter, 19
7 Ein Ogerwald, 11	Schäfchen Gras, 20
8 Von den Raben, 12	

Alle Zitate 1 Zeile 1, 21	Sitz Eine schlaue Schnecke im Sonnenuntergang. Wie ein Ärger die Welt verdeckt dem Sonnenfeind. Die Anger, wie aus Rossequier, in Glasperl mit Fener Und als das Licht dem Stern entzogt ist es sie, die es erweint.
Hello Welt Hello Welt, Wie geht es dir? Nicht so gut, aber und doch bei mir ich hab dich gern, und bin doch hier.	Das Gedicht ist als Textgedicht entstanden um das Gedichtmodell zu überspielen. Es ist klar, aus wichtigen Tropengesang. ●
Florian Sihler 2019	Florian Sihler 2019
Tone Mit dem Klang der ersten Note wirkt sie so gruell. Als Tänzer erscheint Licht und Schatten, vertieft im Lübeospiel. Du denkst und springst im Takt der Geige, die dich durch die Blüme trägt. Man merkt, wie mit jeder Seite, dass Herz wie das der Harfe schlägt.	5
Florian Sihler 2019	6
Schattenspiele Flucht. Atom gebreden. Fokus, das Ziel. Auf immer verloren. Leben im El. Leben als Sturm. Leben ohne Rhythmus, ohne Melodie, ohne Rhythmus.	Schattenspiele. Als Alltag aus Kristallen. Sind wie doch ewig dem Dunkel verfallen. Wir, die Schatten der Nacht. Sind stille, ohne Rhythmus, ohne Rhythmus.
Florian Sihler 2019	Florian Sihler 2019
Pirsch Hohes Geweih, rote Gefieder, Es kommt sein Lied, immer wieder. Ein Dach aus Wolken, ein Fluss aus Wänden, Und doch ewig an diese Welt gehenden.	Der steife Halt, der starke Blick, Nimmt schaut zu ihnen zurück. Aber es folgt den Klängen, Dort wo sie nicht sind, hören sie nur.
Florian Sihler 2019	Florian Sihler 2019
Langsam und träge, es ist auf der Jagd. Schleicht seine Wege, summmt einsam. Und sie hören nichts, was es sei. Selbst wenn es ruhe, den Tag noch.	Klimpert um den Noten, an der Melodie, Bis sie verschwinden, als gäbe es sie nie. So kält das Wosen, zu dem sie kam, Und irgendwo im Wald, singt sie mit anderen zu singen oft.
Florian Sihler 2019	Florian Sihler 2019
Klang Und auch wenn die Hörer nur ich bin, Und ich nur ein Hörer, der mich nicht hört, Hat du der Aderpunkt in meiner Welt die übertratet, Und verhindert, Tag für Tag, dass sie dem Ruhefeld nachgleit.	Dass Blumen ist das Notenmuster, ohne Wann und wo, Dass Blumen ist das Notenmuster, ohne Wann und wo, Mit dem Klang der ersten Note, wirkt sie so gruell. Und deine Welt hältst du, hältst mich am Leben, Auch wenn die Nachschlag schon lange ist.
Florian Sihler 2019	Florian Sihler 2019
Und ich dachte dir dafür... Danke	Und auch wenn die Hörer nur ich bin, Und ich nur ein Hörer, der mich nicht hört, Hat du der Aderpunkt in meiner Welt die übertratet, Und verhindert, Tag für Tag, dass sie dem Ruhefeld nachgleit.
Florian Sihler 2019	Florian Sihler 2019

9.6 Titelseiten

Diese Definitionen werden über keine gemeinsame Bibliothek zur Verfügung gestellt, da sie sich unter Umständen sogar gegenseitig ausschließen können. Lilly lädt lediglich `LILLYxRANDOMxFLAVOURTEXT` und `LILLYxPHILOSOPHER`.

9.6.1 Zufällige Texte

Diese Definitionen werden über die Bibliothek `LILLYxRANDOMxFLAVOURTEXT` zur Verfügung gestellt. Dieses Paket nutzt `LILLYxRANDOM` und `LILLYxENCODING` und definiert einige Gedichte die als `\LILLYxFlavourText` verwendet werden können.

◊ \RandomFlavourText

v2.0.0

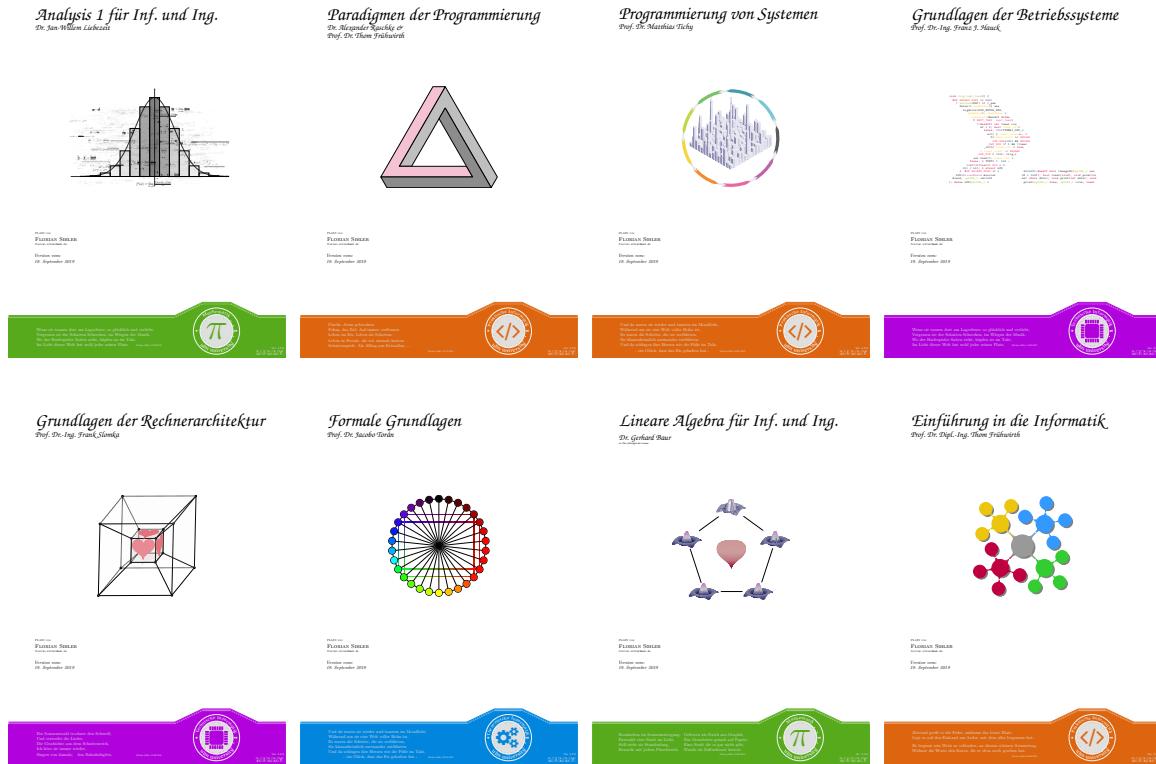
Liefert einen zufälligen Text. Beispiel: `\RandomFlavourText` liefert (in kleiner Schrift :D):
 Die leeren Augen durchbohren dich.
 Sie ist Tod, Doch wie sie starb,
 Das weißt du nicht.
 Die Raben haben sie gefunden und sie geschunden.
 Doch sind mit deinem Anblick, zurück,
 in ihre Welt verschwunden...

Florian Sihler, 04.07.2016

9.6.2 Philosopher

Diese Definitionen werden über die Bibliothek `LILLYxPHILOSOPHER` zur Verfügung gestellt.

Dieses Paket hat mit `VER 1.0.8` die Generierung der Titelseiten für Mitschriften und mit `VER 2.0.0` auch für Zusammenfassungen übernommen. Hier ein kleines Beispiel:



◊ \@@@university@name

v2.0.0

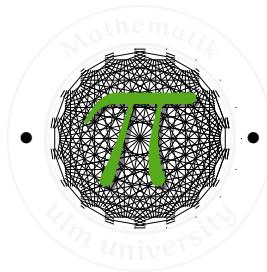
Enthält den Namen der Universität (momentan ulm university).

- ◊ `\LILLYxGENxFACULTY{Upper Text}{Symbol}{outer color}` v1.0.8
`[symbol Color=<black>][tikz args][symbol size=<80pt>]`
`[font opts upper][offset text=<0.1>][offset shadow=<0.05>]`

Ein Befehl, der aus offensichtlichen Gründen wohl nicht wirklich frei verwendet werden sollte. Stattdessen sollte sich das folgende Template genauer ansehen und auf der Basis sein eigenes kreieren:

```
1 \DeclareDocumentCommand{\LILLYxFACULTYxMATHE}{ O{MudWhite} O{%
2   FacultyMathexColor} O{} }{%
3   \LILLYxGENxFACULTY{Mathematik}{$\wp_i$}{#1}[#2][#3][80pt][%
4     \smallNumber][0.125]
5 }
```

Dies definiert das Mathe-Siegel, `\LILLYxFACULTYxMATHE` ergibt (wird zusätzlich zentriert):



- ◊ `\LILLYxFACULTYxMATHE[White=<MudWhite>][Color=<FacultyMathexColor>]` v1.0.8
`[tikz Args]`
- ◊ `\LILLYxFACULTYxPRAKTISCHEINFORMATIK[White=<MudWhite>]` v1.0.8
`[Color=<FacultyPraktischeInformatikxColor>][tikz Args]`
- ◊ `\LILLYxFACULTYxTHEORETISCHEINFORMATIK[White=<MudWhite>]` v1.0.8
`[Color=<FacultyTheoretischeInformatikxColor>][tikz Args]`
- ◊ `\LILLYxFACULTYxTECHNISCHEINFORMATIK[White=<MudWhite>]` v1.0.8
`[Color=<FacultyTechnischeInformatikxColor>][tikz Args]`

Diese Befehle werden verwendet um die jeweiligen Fakultätssymbole zu setzen. Das wirkt erstmal sehr verwirrend und kompliziert. Es ist allerdings relativ einfach, für eine Vorlesung (oder ein Dokument) die Daten zu setzen:

```
1 \def\LILLYxFACULTY{\LILLYxFACULTYxMATHE} % Fakultätssymbol
2 \def\LILLYxFACULTYxCOLOR{FacultyMathexColor} % Fakultätsfarbe
```

- ◊ `\LILLYxColorxTITLExSETTINGSxGENERAL,` v1.0.9
`\LILLYxColorxTITLExSETTINGSxVORLESUNG`
- Diese Befehle definieren die Pfade zu den entsprechenden Definitionen. Sie werden jeweils nur dann eingebunden, wenn `\LILLYxSemester` einen Wert größer 0 hält. Hier sind die Standartdefinitionen:

```

1 \providecommand{\LILLYxColorxTITLExSETTINGSxGENERAL}{%
2   \LILLYxPATHxDATA/Semester/\LILLYxSemester/Definitions/GENERAL.tex%
3 }
4 \providecommand{\LILLYxColorxTITLExSETTINGSxVORLESUNG}{%
5   \LILLYxPATHxDATA/Semester/\LILLYxSemester/Definitions/%
6   \LILLYxVorlesung%
7 }

```

- ◊ **\LILLYxPHILOSOPHERxBORDERBLOCK** [*Signature Color=⟨\LILLYxFACULTYxCOLOR⟩*] v1.0.9
 [Text color=⟨LightGray⟩] {Baseheight}
 [Flavour Text=⟨\LILLYxFlavourText⟩] [Symbol=⟨\LILLYxFACULTY⟩]

Setzt den unteren Block einer Titelseite. Normalerweise wird dieser über die gesamte Breite der Seite gesetzt, hier wurde das ganze natürlich runterskaliert:

```

1 \def\LILLYxFACULTY{\LILLYxFACULTYxMATHE}
2 \def\LILLYxFACULTYxCOLOR{FacultyMathexColor}
3 \resizebox{\linewidth}{!}{%
4   \LILLYxPHILOSOPHERxBORDERBLOCK{3}
5 }

```

Liefert (die Generierung eines Flavour Texts durch zum Beispiel **LILLYxRANDOMxFLAVOURTEXT** wurde deaktiviert!):



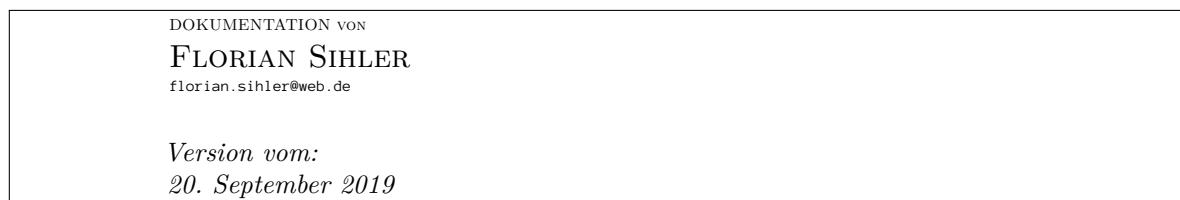
Wie zu sehen ist, setzt **\LILLYxPHILOSOPHERxBORDERBLOCK** die Hautpfarbe des Fakultäts-symbols auf Weiß (genauer Mudwhite).

- ◊ **\LILLYxPHILOSOPHERxINIT** v1.0.9

Wird vom Paket verwendet um die Titelseite zu initialisieren.

- ◊ **\LILLYxPHILOSOPHERxMETADATA** v1.0.9

Setzt die Metadaten wie den Autor(**\AUTHOR**) und dessen Emailadresse (**\AUTHORMAIL**). Hier ein Beispiel (der Rahmen wurde explizit hinzugefügt): **\LILLYxPHILOSOPHERxMETADATA** liefert:



- ◊ **\LILLYxPHILOSOPHERxBONUSxTTOCxHEADER** v1.0.9

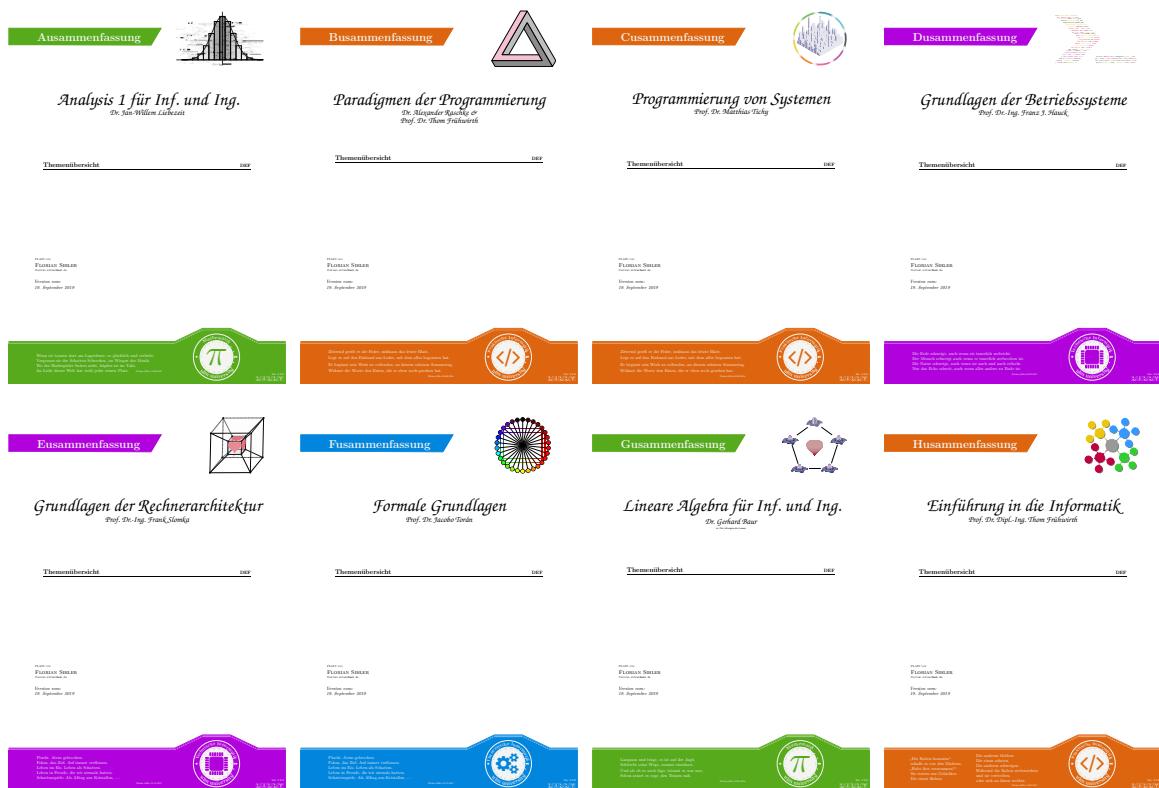
Setzt den Titel für eine Themenübersicht auf der Titelseite, wie sie zum Beispiel das **Zusammenfassungs-Layout** setzt.

- ◇ **\LILLYxTITLExBONUS** [*Signature Color=***\LILLYxFACULTYxCOLOR**] [*Text color=MudWhite*] [*Text*] [*Flavour Text=***\LILLYxFlavourText**] [*Symbol=***\LILLYxFACULTY**] [*Scaling=***0.55\paperwidth**] [*Titlesymbol=***\LILLYxPATHxDATA/Semester/...**]

Setzt die Titelseite wie sie das **Zusammenfassungs-Layout** setzt. Der Text wird links oben als Bonus gesetzt. Hier ein Beispiel:

```
1 \def\LILLYxFACULTY{\LILLYxFACULTYxMATHE}
2 \def\LILLYxFACULTYxCOLOR{FacultyMathexColor}
3 \def\LILLYxSemester{2}\def\LILLYxVorlesung{ANA1}
4 \LILLYxTITLExBONUS{Hallo}
```

Es liefert ein bekanntes Ergebnis auf der folgenden Seite. Zuvor, erst einmal die Übersicht über die für gewöhnlich generierten Zusammenfassungen:



Hallo

Analysis 1 für Inf. und Ing.

Dr. Jan-Willem Liebezeit

Themenübersicht

DEF

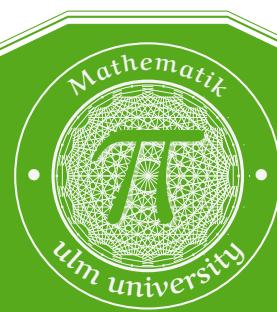
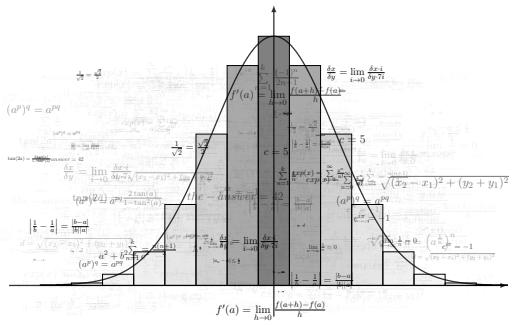
DOKUMENTATION VON

FLORIAN SIHLER
florian.sihler@web.de

Version vom:
20. September 2019

Ein Sonnenstrahl trocknet den Schweiß,
Und vertreibt die Lieder.
Die Geschichte aus dem Schattenreich,
Ich höre sie immer wieder,
Singen von damals; Am Bahnhofsgleis.

Florian Sihler 12.08.2016



Offensichtlich wird `\LILLYxPHILOSOPHERxBORDERBLOCK` verwendet.

◊ `\LILLYxTITLExRAW[Title Image PDF]`

v1.0.9

Setzt eine Titelseite *ohne* Bonus, so wie sie bereits in der [Einführung](#) gezeigt wurden. Das anzugebende Titelbild sollte übergeben werden. Es wird auf `\LILLYxPHILOSOPHERxBORDERBLOCK` und `\LILLYxPHILOSOPHERxMETADATA` zurückgegriffen.

◊ `\LILLYxTITLE`

v1.0.9

Setzt die Titelseite mittels `\LILLYxTITLExRAW`, allerdings nur, wenn `\LILLYxSemester` größer 0 ist. In diesem Fall wird allerdings automatisch das dazugehörige Titelbild hinzugefügt.

10

JAKE

Jake! Would you get me the cake please?...

VER 1.0.8

10.1 Grundlegendes

10.1.1 Entwicklung

Anfänglich wurde *Jake* als *installer* konzipiert, der einfach nur die mühselige Installation des Pakets abnehmen soll. Mittlerweile hat sich *Jake* allerdings weiterentwickelt und bietet das Potenzial für einiges mehr. Im Folgenden sei die Funktionsweise genauer erklärt. Zu beachten ist allerdings, dass *Jake* bisher nur für Linux und MacOS einen Installer und somit seine Funktionalität zur Verfügung stellt!

10.1.2 Die Installation

Jake wird als .jar-Datei geliefert und lässt sich, eine vorhandene Installation von Java vorrausgesetzt, durch das bloße ausführen installieren. Auf Linux kann dies zum Beispiel wie folgt von statteten gehen:

```
java -jar jake.jar
```

Nach abgeschlossener Installation sollte das Terminal neu gestartet, oder die Konfigurationsdatei neu geladen werden, um *Jake* zur Verfügung zu stellen. Das bloße Ausführen von *jake* sollte nun eine Hilfe anzeigen, die über die jeweiligen Optionen aufklärt.

Bemerkung 31 – C++ Jake

Bis zur Version VER 1.0.9 war Jake in C++ geschrieben und benötigt deswegen eine andere Installation.

Jake zu installieren sollte normalerweise einem Kinderspiel gleichen. Notwendig sind hierfür auf allen bisher unterstützten Betriebssystemen (Debian-Basiertes Linux und MacOS) ein C++14 fähiger gcc-Compiler und make. Anschließend gilt es ins *jake_source*-Verzeichnis zu navigieren. Es befindet sich hier: *Lilly/Jake/jake_source*. In diesem Verzeichnis kann man nun make ausführen. Dies sorgt dafür, dass nicht nur *jake.cpp* zu einer ausführbaren Datei wird, sondern auch, dass *lilly_jake* systemweit zur Verfügung steht (sofern die verwendeten Konsole bash, zsh oder iTerm ist, bzw. im allgemeinen auf eine der folgenden Dateien zugreift: *.bashrc*, *.zshrc*, *.bash_profile*). Damit gilt *Jake* als *installiert*.

10.1.3 Lilly mit Jake installieren

Mit *VER 2.0.0* liefert *Jake* stets eine Version von Lilly mit, die sich nach einem Update automatisch mit dem nächsten Start von *Jake* aktualisiert, sofern sie einmal installiert wurde: *jake install*. Wird hier eine Frage nach verschiedenen Installationsoptionen gestellt, so

siehe bei den [Entwicklerinformationen](#) oder wähle einfach die Installation der enthaltenen Variante (vermutlich Option 2). Die automatische Aktualisierung wird durch eine Ausgabe getreu [Die Lilly-Installation wurde aktualisiert.] ausgegeben. Mithilfe von [jake GUI](#) kann Jake auch über die Kommandozeile im Grafischen-Modus gestartet werden, allerdings empfiehlt sich hierfür das Verwenden des Eintrags im Anwendungsmenü.

10.1.4 Jake im Überblick

Hier werden zuerst die Vorteile der Kommandozeile präsentiert, da die grafische Variante von Jake noch nicht sinnvoll ausgebaut ist und sich bisher lediglich zum editieren von Konfigurationsdateien eignet.

Kommandoszeile

Im Regelfall, zur Kompilierung eines Dokuments, genügt es [Jake](#) mit dem jeweiligen Dokument aufzurufen:

```
jake <Dokumentname.tex>
```

Der Kompilierung können nun eine endlose Reihe an Einstellungen übergeben werden die jeweils mit einem „-“ anzuführen sind. Eine boolesche Einstellung kann so bereits umgeschaltet werden. So liefert:

```
jake dump --debug --debug --debug
```

Für die Einstellung „debug“ *true*. Eine „normale“ Einstellung, welche ein Argument fordert wird durch einen Doppelpunkt beendet:

```
jake dump -lilly-author: "Sonnenprophet_Hamsterbacke"
```

Liefert den entsprechenden Author für *lilly-author*. Dies lässt sich auch bei booleschen Ausdrücken, hier mit dem Setzen der Werte *true* und *false* erzeugen. Final gibt es noch Listen, die auch so zugewiesen werden können, allerdings durch das anfügen von +: auch erweitert werden können. So liefert:

```
jake dump -lilly-boxes: "DEFAULT" -lilly-boxes+: "ALTERNATE"
```

Den Wert „DEFAULT ALTERNATE“ für die Einstellung *lilly-boxes*, die Trennung der Elemente (Leerfeld) wird von Jake automatisch erkannt ist aber in der Regel auf Leerfelder normiert.

Hier die große (und hoffentlich vollständige) Liste aller möglichen Einstellungen. Ist der Standardwert zu lang, so wird er durch ... gekürzt, wenn er abhängig ist, wird dies in der Bemerkung erklärt. Es gilt zu beachten, dass sich durch Konfigurationsdateien alle Einstellungen modifizieren lassen und somit auch die Standardwerte verändern:

Bezeichner	Typ	Defaultwert	Beschreibung
Version	<i>String</i>	[...]	Aktuelle Version von Jake
file	<i>String</i>	dummy.tex	Datei, um die es gehen soll
answer	<i>String</i>		Antwort, die, sofern nicht leer, auf alle Fragen die Jake stellt zuerst gegeben wird. Ein setzen auf „y“ entspricht der -y-Option von apt.

operation	<i>String</i>	help	Was Jake tun soll
debug	<i>Boolean</i>	false	Gibt an, ob Debug ausgegeben werden soll oder nicht
debug-filter	<i>String</i>	.*	Veraltet
path	<i>String</i>	./	Pfad zu Lilly
what	<i>String</i>		Zusatzargument für manche Operationen
install-path	<i>String</i>	\$HOME/texmf	Ziel Pfad der Installation
gepardrule-path	<i>String</i>		Pfade für Gepardregeln (durch „;“ getrennt)
autoconf	<i>Boolean</i>	true	Soll automatisch eine .conf-Datei gewählt werden?
comment-pattern	<i>String</i>	! [^!]*!	Kommentarmuster
<hr/>			
lilly-path	<i>String</i>	\$(dirname...)	Pfad zur <code>Lilly.cls</code>
lilly-out	<i>String</i>	./\$(BASE...)	Ausgabeordner der Tex-Datei?
lilly-in	<i>String</i>	./	Input-Pfad für Dateien
lilly-nameprefix	<i>String</i>		Namenspräfix für Ausgabedatei
lilly-boxes	<i>List</i>	DEFAULT	Boxen für den Kompilervorgang
lily-modes	<i>String</i>	default	Modi für den Kompilervorgang
lilly-complete	<i>Boolean</i>	true	Vollständige Dokumentvariante
lilly-complete-name	<i>String</i>	COMPLETE-	Präfix der vollständigen Version
lilly-print-name	<i>String</i>	PRINT-	Präfix der Druckversion
lilly-cleans	<i>List</i>	log aux ...	Dateiendungen die von autoclean gelöscht werden
lilly-autoclean	<i>Boolean</i>	true	Sollen Dateien automatisch gelöscht werden?
lilly-compiletimes	<i>String</i>	2	Wie oft soll kompiliert werden
lilly-vorlesung	<i>String</i>	NONE	Um welche Vorlesung handelt es sich?
lilly-semester	<i>String</i>	0	Das wievielte Semester ist es
lilly-n	<i>String</i>	42	Um das wievielte Übungsblatt handelt es sich?
lilly-show-boxname	<i>Boolean</i>	true	Soll der Boxname angezeigt werden?
lilly-layout-loader	<i>String</i>		Pfad zu den Layouts

lilly-external	<i>Boolean</i>	false	Soll versucht werden, Grafiken auszulagern?
lilly-external-out	<i>String</i>	extimg	Ausgabeordner für ausgelagerte Grafiken
lilly-author	<i>String</i>	Florian...	Author des Dokuments
lilly-author-mail	<i>String</i>	florian.s...	Email-Adresse des Authors
lilly-signatur-farbe	<i>String</i>	Leaf	Farbe für das Highlighting
lilly-bibtex	<i>String</i>		Bibtex-Datei (ohne Endung)
lilly-doctype	<i>String</i>	Mitschrieb	Typ des Dokuments
lilly-configs-path	<i>String</i>		Pfad zur Lilly-Konfigurationsdatei (\lillyPathConfig)
lilly-data-path	<i>String</i>		Pfad zu generellen Daten (\lillyPathData)
jobcount	<i>String</i>	2	Wie viele verschiedene Threads sollen im multithreaded-compile gleichzeitig betrieben werden?
error-count	<i>String</i>	5	Wie viele Fehler sollen in der Vorschau maximal angezeigt werden?
mk-name	<i>String</i>	Makefile	Veraltet, Name des Makefiles
mk-path	<i>String</i>	./	Veraltet, Pfad des Makefiles
mk-use	<i>Boolean</i>	false	Veraltet, soll das Makefile verwendet werden?

Zusätzlich kann anstelle der vorangestellten Option wie `dump` beziehungsweise der `.tex`-Datei auch eine Konfigurationsdatei angegeben werden. Auf sie wird [hier](#) mehr eingegangen. Bei einer solchen Angabe handelt es sich um eine Kurzform der jeweiligen Optionen `config` und `file_compile`. Ausführlich würde man also zum Kompilieren dieser Dokumentation schreiben:

```
jake file_compile -file: Lilly-Dokumentation.doc.tex
```

beziehungsweise, zum Verwenden der beiliegenden Konfigurationsdatei:

```
jake config -file: doc.conf
```

Oder eben die Kurzform:

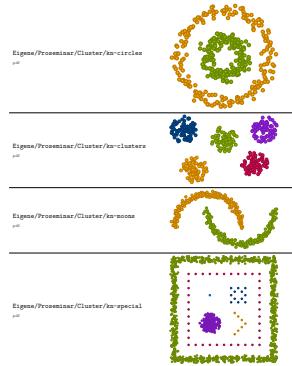
```
jake doc.conf
```

Die automatisch die Optionen entsprechend setzt.

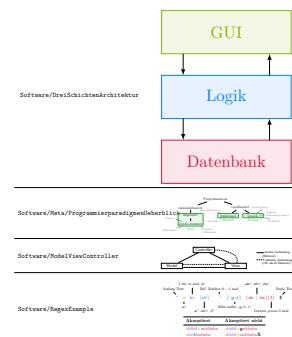
Als letzte wichtige Funktion sei noch `get` genannt, welche für das Paket `LILLYxGRAPHICSxPROVIDER` aus `LILLYxGRAPHICS` relevant ist. So liefert der Befehl:

```
jake get
```

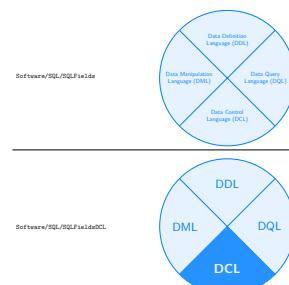
Eine PDF wie die folgende, die alle mit Lilly gelieferten Grafiken enthält:



13



33

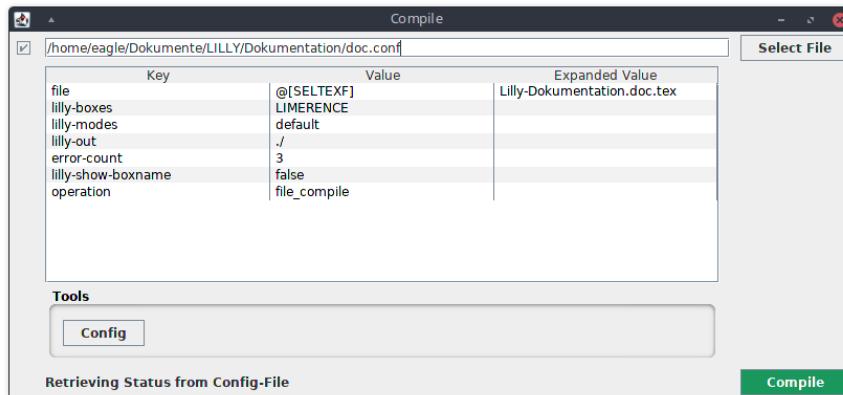


34

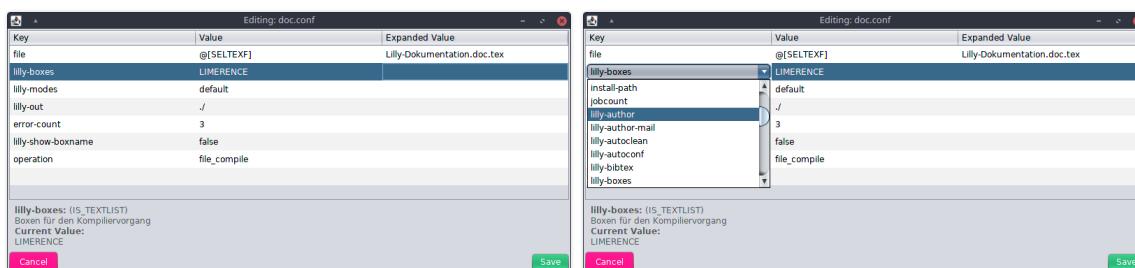
Angezeigt wurden hier übrigens die Seiten 13, 33 und 34 zum Kompilierzeitpunkt dieser Dokumentation.

Gui

Wie bereits angemerkt, ist die GUI von *Jake* noch nicht ansatzweise ausgereift. Der sich öffnende Hauptdialog erlaubt das Auswählen einer Latex- oder Konfigurationsdatei und zeigt die aus der Datei extrahierten Informationen inklusiver ihrer (sofern verschieden) extrahierten Werte an:



Nebst einem schönen Ausblick auf das was in der GUI-Welt noch so alles geschehen mag, bietet sich der Button **Config** an, der im Falle keine ausgewählten Konfigurationsdatei einfach auf einer neuen arbeitet:



Hier werden nicht nur einige Informationen zu den jeweileigen Einstellungen angezeigt, sondern auch Plausibilitätsprüfungen durchgeführt. Im Falle einer neuen Konfigurationsdatei kann beim Speichern ein entsprechendes Ziel ausgewählt werden.

10.1.5 Entwicklerinformationen

Wenn du an *Jake* oder Lilly mitentwickelst, oder einfach generell immer die aktuellste Version von Jake haben möchtest, so gilt es ein paar Schritte zu befolgen. Vorab: *Jake* wurde als Maven-Projekt angelegt, zum generieren der neusten Version ist also Maven für die jeweilige Plattform vonnöten (Beispiel: `sudo apt install maven`)

1. Klone das Git-Repository (<https://github.com/EagleoutIce/LILLY>)
2. Navigiere in das Verzeichnis *Jake/* im Repository
3. Generiere die neuste *Jake*-Version:

```
mvn clean install
```

4. Wähle aus (beachte die Angabe einer **Nutzerkonfiguration**):

stable Diese Installation wird unregelmäßig aktualisiert und entspricht der *jake.jar*, muss also nicht extra gebaut werden. Diese Variante empfiehlt sich für Tests und für die Entwicklung an Lilly, da so Fehler auf Seiten von *Jake* in der Regel ausgeschlossen werden können.

rolling Diese Installation wird nicht über das git-Repository synchronisiert sondern kann vom Nutzer bei Bedarf erzeugt werden. Hierzu einfach im *Jake*-Verzeichnis, den obigen Befehl (`mvn clean install`) ausführen. Im Unterverzeichnis *target/* befinden sich danach die *development-jake.jar*, die nach dem Ausführen mit jedem weiteren Kompilieren von *Jake* automatisch aktualisiert:

```
java -jar development-jake.jar
```

5. Bei der Installation von Lilly mit `jake install`^{a)} wird nun sicher eine Frage erscheinen, welche Variante von Lilly installiert werden soll. Während im **stable-tree** beide Optionen theoretisch verwendbar sind, so empfiehlt sich - auch um immer die aktuellste Lilly-Version zu haben, die Verlinkung der gefundenen Lilly-Instanz. Hier allerdings Vorsicht, da der Pfad zur *Lilly.cls* falsch sein kann. Der Pfad sollte in seiner Signatur auf *LILLY/Lilly/Lilly.cls* enden. Ist dies nicht der Fall, so muss die Option *lilly-path* angegeben werden, die den (am besten absoluten) Pfad zur *Lilly.cls* angibt.

10.2 Gepard

Die **Generator-Parser-Descriptor-Language** ist die Sprache, in der alle Konfigurationen und Erweiterungen von Jake formuliert sind. Im Kern des Parsen von Dokumenten steht der **Tokenzier** der erlaubte Zeichen und Zuweisungen unterscheidet, und vom **Configurator** erweitert wird. Dieses Konfigurationsmodul gestattet die Definition von **Konfigurationsdateien**, die durch einfache Zuweisungen die Einstellungen von *Jake* kontrollieren können. Darüber baut das namensgebende Gepard-Modul auf, welches verschiedene Boxen und damit

^{a)} Nicht vergessen, dass Terminal neu zu starten/die Konfigurationsdatei der Konsole neu einzulesen.

verschiedene Erweiterungen gewährt, wobei bisher ein verschachteln dieser Boxen nicht vorgesehen ist. Die einzelnen Module werden weiter unten beschrieben.

Bemerkung 32 – Kommentare

Ein Kommentar in Gepard wird in der Regel durch Ausrufezeichen markiert. Sollten diese allerdings verwendet werden müssen, so ist es möglich die Sequenz für Kommentare mithilfe von `comment-pattern` zu modifizieren:

```
1 debug    = true
2 ! debug = false !
3 what     = /* Noch kein Kommentar */
4 comment-pattern = /*.**/
5 lilly-author   = Hallo ! Sonne !
6 answer   = 42, /* Ich bin jetzt ein Kommentar :D */
```

Wir erhalten (mit `dump`) die Ausgabe für die modifizierten Werte:

```
comment-pattern      : [/*.**/
debug               : [true]
lilly-author        : [Hallo ! Sonne !]
what                : [/* Noch kein Kommentar */]
answer              : [42,]
```

Keine mit Jake gelieferte Konfigurations- oder anderweitige Datei modifiziert die Syntax für einen Kommentar.

10.2.1 Konfigurationsdateien

Eine Konfigurationsdatei endet für gewöhnlich auf `.conf`, wobei diese Endung lediglich von der Auto vervollständigung und der GUI anerkannt wird, allerdings keineswegs verpflichtend ist. *Jake* versucht jede Konfigurationsdatei entsprechend zu parsen. Erlaubt werden alle auch für *Jake* in der Kommandozeile verwendbaren **Einstellungen**, wobei zur Zuweisung hier `=` und `+=` anstelle von `:` und `+`: verwendet werde und kein „`-`“ angeführt wird. So kann eine Konfigurationsdatei wie folgt aussehen:

```
1 operation      = file_compile
2 file          = @[SELTEXF]
3 lilly-modes    = default
4 lilly-show-boxname = false
5 lilly-boxes    += LIMERENCE
6 lilly-out       = ./
7 error-count    = 3
```

Auch wenn hier zur Optik die Zuweisungen alle auf die gleiche Einrückung gesetzt wurden, so ist dies nicht zwingend und auch Tabs und Leerfelder haben im Verhältnis zur Zuweisung keine semantische Bedeutung und sind auch syntaktisch irrelevant. Das hier enthaltene `@[SELTEXF]` ist ein **Expandable**, welches über ein weiteres Gepard-Modul definiert wird. Dieses evaluiert zur ersten TeX-Datei die im Ordner gefunden wird, hat also den Vorteil, dass diese Einstellung der konfigurationsdatei nicht immer wieder angepasst werden muss. Es gibt einige derartige Einstellungen.

Bemerkung 33 – Setzen von `operation`

Es ist zwangsläufig zu empfehlen die Einstellung `operation` auf das gewünschte Ziel zu überschreiben, da sonst die neue Datei (sofern überhaupt eine andere angegeben wurde) wieder mit der `config`-Opteration ausgeführt wird, was im Zweifelsfall zu einer Endlosrekursion führen kann (diese wird von `Jake` natürlich erkannt und abgebrochen).

Weiter besitzt `Jake` die `Einstellung autoconf`, die eine Konfigurationsdatei bei der Wahl einer TeX-Datei auch automatisch auswählen kann sofern diese den gleichen Namen oder den Namen `jake.conf` trägt. So wird zum Beispiel beim Kompilieren von `Dokument.tex` automatisch die Datei `Dokument.conf` als Konfigurationsdatei geladen, sofern diese existiert. Analog würde die `jake.conf` gewählt werden, wenn sie existiert.

Bemerkung 34 – Standartkonfigurationsdateien

`Jake` selbst kommt mit der `jake_default.conf`, das ist eine Konfigurationsdatei, die für den aktuellen Build Einstellungen setzt, ohne jedesmal die in den `CoreSettings` vermerkten Einstellungen zu modifizieren. Diese Datei lässt sich theoretisch problemlos anpassen, davon wird allerdings stark abgeraten, da derartige Modifikationen mit einer Aktualisierung von `Jake` wieder überschrieben werden. Allerdings kann bei der Installation von `Jake` die Einstellung `path` gesetzt werden um eine Nutzerkonfiguration anzugeben. Diese wird von da an immer beim Starten von `Jake` eingelesen und verarbeitet:

```
java -jar jake.jar -path: /pfad/zu/meiner/Konfiguration.conf
```

Eine vorhandene `Jake`-Installation (auch zum Abändern dieses Pfades) kann mithilfe von `jake DEI` deinstalliert und mit `jake REI` reinstalled werden, so kann auch bei einer bestehenden Installation von `Jake` mithilfe von:

```
jake REI -path: /neuer/pfad/zu/meiner/Konfiguration.conf
```

Der Pfad aktualisiert und mit:

```
jake REI
```

Der Nutzerpfad gelöscht werden. So Kann man mit dieser Konfiguration den Author aller Dokumente auf sich verändern:

```
1 ! Setze Autor für alle Dokumente: !
2 lilly-author      = Kalle Uweson
3 lilly-author-mail = kalle.uweson@hotmail.waffle
4 ! Verstecke standardmäßig den Boxnamen: !
5 lilly-show-boxname = false
6 ! Setze standardbox auf ALTERNATE !
7 lilly-boxes       = ALTERNATE
```

Aktuell wird überlegt, ob bei der Instalation direkt nach wichtigen Daten wie dem Namen gefragt wird.

10.2.2 Gepard Module im Allgemeinen

Gepardmodule werden in einer Datei als Box präsentiert. Eine Datei kann so etliche verschiedene Boxen und damit Konfigurationen für verschiedene Module halten und verarbeiten. Das grundlegende Gepard-Modul kann in einer (üblicherweise auf .gpd endenden) Datei wie folgt dargestellt werden:

```
1 BEGIN <modul>
2   <Modulspezifikation>
3 END
```

In die jeweiligen Start- und Endzeilen können beliebige Zeichen zur Übersicht Platziert werden, sie werden verworfen, was zum Beispiel folgende Spezifikation genauso valide macht:

```
1 BEGIN <modul>:
2   <Modulspezifikation>
3 END;
```

Die Spezifikation besteht in der Regel aus Konfigurationsähnlichen Zuweisungen, die je nach Modul eine unterschiedliche semantische Bedeutung haben. Die gewünschten Konfigruationen können über die Einstellung `gepardrule-path` gesetzt werden, wobei die Pfade durch einen „::“ getrennt sind. Bisher muss vom Nutzer die Existenz der zugrundeliegende Dateien gewährleistet werden.

10.2.3 Buildrules

Buildrules definieren den Modus in dem das Dokument kompiliert wird. So definieren sie den `print` und den `default` Modus. Die Box trägt den Namen `buildrule` und muss einen Namen, einen Anzeigenamen und einen Modus definieren. Im Folgenden die Definition des `default`-Modus, die Kommentare sollten die Anforderungen zu Genüge erklären:

```
BEGIN buildrule: ! Der Doppelpunkt ist optional. Ich mag ihn, man braucht ihn nicht !
! Das Einrücken _und_ die Leerfelder sind optional. !
! Allerdings sollten erstmal nur Leerfelder verwendet werden !
! Mit X sind Zuweisungen markiert die verpflichtend sein sollen (aber nicht sind) !
!X! name          = default      ! buildrule name für lilly-modes !
!X! display-name = Standard     ! Anzeigename (Standard-Version) !
!X! lilly-mode    = default      ! Welcher Modus soll an Lilly übergeben werden ?
! Info: Diese können noch nicht frei konfiguriert werden !
      complete     = false        ! Keine complete-Version !
      complete-prefix = c_          ! Bezeichner wenn complete !
      nameprefix    = MY-DEFAULT- ! Weicht vom normalen default ab !
      lilly-complete-prefix = COMPLETE- ! Namenszusatz wenn complete Version (Default: COMPLETE-) !
      lilly-loader   = \\input{$(INPUTDIR)$(TEXFILE)}
! Diese Funktion ist advanced und beschreibt die Einbinderoutine – einfach ignorieren !
END; ! Semikolon wieder nicht nötig, aber ich mag es :D !
```

Jeder so definierte Modus steht in den Einstellungen für `lilly-modes` zur Verfügung. Auch wenn sie bisher eher eingeschränkt agieren können, so bieten sie bereits einiges an Flexibilität.

10.2.4 Expandables

Die hier definierten Variablen können überall in Einstellungen oder anderen Gepardrule-Files verwendet werden. Abgesehen von einer rekursiven Definition ist alles gestattet. Jake definiert

bereits eine Reihe an Expandables, ein paar davon greifen auf Shell-Befehle zurück, was aus Sicherheitsgründen sonst nicht gestattet ist (im Klartext: Auch wenn es vordefinierte Expandables gibt die auf Shell-Befehle zurückgreifen, kann kein manuell definiertes Expandable eigene Shell-Befehle inizieren). Im Folgenden sind jeweils nur ihre Bezeichner angegeben, jedes Expandable kann durch `$[<Name>]` und `${<Name>}` angegeben werden um zum Zielwert zu evaluieren:

Bezeichner	Evaluiert zu
TEXFILE	Expandiert zum vollen Bezeichner TeX-Datei
BASENAME	Expandiert zum Namen der TeX-Datei ohne Endung
FINALNAME	Expandiert zum Namen nach der Generierung (nur sofern im Kontext klar vorhanden)
LOGFILE	Expandiert zum Pfad der Logdatei
PDFFILE	Expandiert zum Namen der PDF-Datei
LATEXARGS	Expandiert zu den Latex-Argumenten (<code>-shell-escape</code> , ...)
OUTPUTDIR	Expandiert zum Ausgabeordner
INPUTDIR	Expandiert zum Quellordner
BOXMODES	Expandiert zu den Boxmodi
CLEANTARGETS	Expandiert zu den zu löschen Endungen
SIGNATURECOL	Expandiert zur Signaturfarbe
AUTHOR	Expandiert zum Author
AUTHORMAIL	Expandiert zur Email-Adresse des Autors
NAMEPREFIX	Expandiert zum Namenspräfix
SEMESTER	Expandiert zur Semesterzahl
VORLESUNG	Expandiert zur Vorlesung
LILLY_CONFIGS_PATH	Expandiert zum Pfad der Konfigurationen
LILLY_DATA_PATH	Expandiert zum Pfad der Daten
N	Expandiert zur Übungsblattnummer
JOBCOUNT	Expandiert zur Maximalen Jobanzahl
_LILLYARGS	Expandiert zu den Argumenten für Lilly
_C	Expandiert zu einem wundervollen Komma ☺
HOME	Expandiert zum Homeverzeichnis
TRUE	Expandiert zu „true“
FALSE	Expandiert zu „false“

S_TRUE	Expandiert zur Jake-Definition von <i>true</i> („true“)
S_FALSE	Expandiert zur Jake-Definition von <i>false</i> („false“)

Es existieren noch einer Reihe besonderer Expandables, die entweder Shell Befehle beinhalten, oder außerhalb des direkten Dokumentenkontext steht. Sie besitzen die Signatur @[<Name>] und sind in der Regel lazy:

Bezeichner	Evaluiert zu
JAKEVER	Expandiert zur Jake Version
SELTEXF	Expandiert zu einer TeX-Datei des Verzeichnisses
SELCONF	Expandiert zu einer .conf-Datei des Verzeichnisses
GITHUB	Expandiert zum Github-Link des Repositories
CONFPATH	Expandiert zum Pfad der Nutzerkonfiguration
AUTONUM	Expandiert zu einer Zahl im Dateinamen, sofern dieser eine Zahl enthält, sonst 42
WAFFLE	Expandiert zur „GIVE ME THAT WAFFLE“ und wird für Tests verwendet
JAKECDATE	Veraltet, ist zum Kompiledatum der C++-Version expandiert
JAKECTIME	Veraltet, ist zum Kompilezeitpunkt der C++-Version expandiert

Die Definition eines Expandables ist relativ einfach, jede Zuweisung der Box wird als Expandable zur Verfügung gestellt:

```

1 BEGIN expandable:
2   SUPERWAFFEL = Ist Wichtig
3   S_TRUE      = FALSE ! Tihihih !
4   S_FALSE     = TRUE ! höhöhöhö !
5   SuperHome   = ${HOME}/Tolle Welt/${TRUE}
6   LayoutConfig = @[CONFPATH]/Layout
7 END;
```

Sie lassen sich normal durch \${<name>} und \${(<name>)} erweitern.

10.2.5 Hooks

Hooks sind etwas tolles ☺, sie können während des Kompilierprozesses Shell-Befehle ausführen und so Aktionen übernehmen wie das Verschieben von Dateien oder dem Anstoßen weiterer Kompilierprozesse. Sie sind es auch, die beim Kompilieren den aktuellen Stand sowie die Lokalität des Logfiles ausgeben. Eine Hook besteht aus den folgenden Komponenten:

```

1 BEGIN hook:
2 !X! name      = in0-hook ! :D !
3 !X! type      = IN0
```

```

4   body      = echo "Hallo Welt – will it break?"
5   on-failure =          ! nothing at all !
6   on-success = ""       ! still nothign !
7 END;

```

Der Typ (type) einer Hook, kann die folgenden Bezeichner annehmen:

Type	Bezeichnung
PRE	Wird vor dem Kompilieren ausgelöst
IN#	Führt, von 0 beginnend die Hook nach dem #-Kompilervorgang aus (IN1, IN42, ...)
POST	Wird nach dem Kompilieren ausgelöst
ALL	Wird jedesmal ausgelöst

Wird die Hook ausgelöst, so wird der `body` in der entsprechenden Shell des Betriebssystems ausgeführt. Im Falle einer geglückten Operation wird `on-failure` im Fehlerfall wird `on-success` ausgeführt.

10.2.6 Name Maps

Um Faul bleiben zu können, wurde das `nmap`-Modul kreiert, welches für den Namen eines Dokuments gewisse Trigger generieren kann, die verschiedene Einstellungen setzen können. So existieren eine Reihe an Name Maps, die im Falle eines entsprechenden Dokumentnamens die jeweilige Einstellungen übernehmen:

Name	Trigger	Erklärung
PDP	pdp,PdP,PDP,[Pp]aradigmen[\\" \\\n]?([Dd]er[\\" \\\n-]?)?[Pp]rogrammierung	Setzt das Semester auf 2 und die Vorlesung auf PDP
GDBS	gdbS,GdBS,GDBS,[Gg]rundlagen[\\" \\\n-]?([Dd]er[\\" \\\n-]?)?[Bb]etriebssysteme	Setzt das Semester auf 2 und die Vorlesung auf GDBS
ANA1	ana1,ANA1,[Aa]nalysis[\\" \\\n-]?1	Setzt das Semester auf 2 und die Vorlesung auf ANA1
PVS	pvs,PvS,PVS,[Pp]rogrammierung[\\" \\\n-]?([Vv]on[\\" \\\n-]?)?[Ss]ysteme	Setzt das Semester auf 2 und die Vorlesung auf PVS
GDRA	[Gg][Dd][Rr][Aa],[Gg]rundlagen[\\" \\\n-]?([Dd]er[\\" \\\n-]?)?[Rr]echnerarchitektur	Setzt das Semester auf 1 und die Vorlesung auf GDRA
EIDI	[Ee][Ii][Dd][Ii],[Ee]inführung[\\" \\\n-]?([Ii]n[\\" \\\n-]?)?([Dd]ie[\\" \\\n-]?)?[Ii]nformatik	Setzt das Semester auf 1 und die Vorlesung auf EIDI
FG	[Ff][Gg],[Ff]ormale[\\" \\\n-]?([Gg]rundlagen	Setzt das Semester auf 1 und die Vorlesung auf FG
LA	LA,LAI1,[Ll]ineare[\\" \\\n-]?[Aa]lgebra	Setzt das Semester auf 1 und die Vorlesung auf LAII

ÜB

UB,uebungsblatt,[Üü]bungsblatt,ÜB

Setzt den Modus auf „uebungsblatt“

Ein **nmap** braucht einen Namen und ein Pattern, mit dem er auslöst. Alle weiteren Einstellungen die Übergeben werden sind die einer **Konfigurationsdatei** und damit der **Einstellungen** von Jake die so übernommen werden. Hier ein Beispiel:

```
1 BEGIN nmap:
2   name          = Paradigmen-der-Programmierung
3   patterns      = pdp,PDP,PdP
4
5   ! Enthält der Dateiname also 'pdp', 'PDP' oder 'PdP' !
6   ! werden folgende Einstellungen angewendet:
7
8   lilly-author    = Schlingelwingel
9
10  lilly-vorlesung = PDP!!
11  lilly-semester  = 2!!
12 END;
```

11

EXKURSE

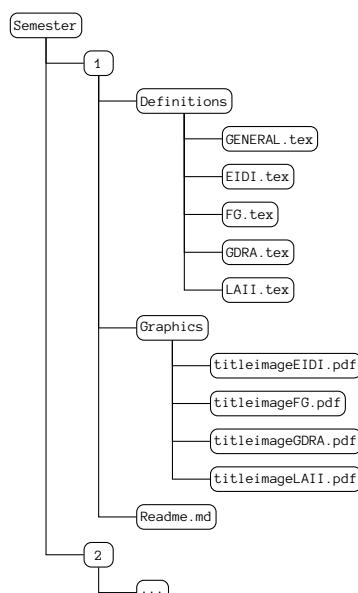
GANZ VIELE VIELE ERKLÄRUNGEN UND BEISPIELE?

11.1 Wie man sich eine eigene Vorlesung bastelt

Bemerkung 35 – Disclaimer

Aktuell kann jake Daten noch nicht wirklich dynamisch laden. Es ist zwar möglich Quellpfade zu ändern, allerdings würde das bedeuten die komplette Ordnerstruktur zu reproduzieren und die notwendigen Standards selbst zur Verfügung stellen zu müssen. Deswegen wird die Vorlesung direkt in Lilly integriert.

Der Datenordner, in dem sich die Vorlesung unter `Lilly/source/Data/Semester/<Semester>/Definitions/<Vorlesung>` befindet, besitzt eine relativ verpflichtende Struktur, die sich geplant mit **VER 2.1.0** ändern soll (das Prinzip eines Semesters soll abgeschafft und das Prinzip einer Vorlesung verallgemeinert werden). Hier die Übersicht der Ordnerstruktur ab `Lilly/source/Data/`:



Die Verzeichnisse können weitere Ordner enthalten diese werden allerdings (wie die `Readme.md`) nicht von Lilly beachtet. Sie können allerdings weitere Informationen für Vorlesungen enthalten, die dann von diesen eingebunden werden, sofern dies benötigt wird. Die Definition `GENERAL.tex`⁽¹⁾ wird immer geladen und kann so für das gesamte Semester Definitionen abgegen (so zum Beispiel das Setzen des `\LILYxKEYVALxPARSER`). Alle anderen Konfigurationen tragen den Namen, der der `\LIB`-Option `Vorlesung` oder der `Jake`-Einstellung `lilly-vorlesung` übergeben werden können. Ein *mapping* dieser Bezeichner findet aktuell nicht statt.

Die Vorlesungskonfiguration

Eine Vorlesungskonfiguration wie `EIDI.tex` kann eine Menge an Befehlen definieren, in der Regel werden allerdings die folgenden gesetzt, wobei die mit `default` markierten Optionen nicht gegeben werden müssen:

⁽¹⁾ Auch bezüglich der Benennung der Konfigurationen ist eine Änderung für **VER 2.1.0** geplant.

Befehl	Erklärung	Beispiel	Notiz
<code>\TITLE</code>	Name der Vorlesungsreihe	Grundlagen der Schäferzucht	

\PROFESSOR	Name des Dozenten	Prof. Dr. Dööst
\UEBUNGSLEITER	Name des Übungsleiters	Max Mustermän
\TUTOR	Name des Tutors	Herr Subertuter
\SUBTITLE	Untertitel	\PROFESSOR Default
\FULLTITLE	Titel für die Titelseite	\TITLE \\\\fontsize{18pt}{16pt} \\selectfont{\SUBTITLE} Default
\UEBUNGSHEADER	Kopfzeile eines Übungsblatts	\TITLE\\Übungsblatt \LILLY@n Default
\VORLESUNG	Vorlesungs-Schriftzug	{ \bfseries Roffledoffel }

Besonders sind die Befehle \UEBUNGSLEITER und \TUTOR, da diese erst dann gegeben werden müssen wenn sie gebraucht werden (genau genommen gilt das für alle Befehle, allerdings werden die anderen in einem Mitschrieb benötigt, um zum Beispiel die Titelseite zu generieren).

Weiter ist es möglich, die beiden folgenden Befehle zu setzen, die eine besondere Bedeutung besitzen:

Befehl	Erklärung	Beispiel
\POLITEINTRO	Einleitung für Zusammenfassungen	Offensichtlich erhebt dieses Dokument keinen Anspruch...
\LILLYxTITLExOffset	Setzt das horizontale Titelbildoffset bei der Zusammenfassung im Falle einer abweichenden Größe	11.6cm

Zur Definition weiterer Einschränkungen wie zum Beispiel der Sichtbarkeit von LIB LILLYxBOXES sollten mit \providedef getätigter werden, können allerdings auch anderweitig gesetzt werden, wenn es zum Beispiel nicht anders funktioniert. Hier ein Beispiel für die Vorlesung ANA1:

```

1 %% Diese Datei enthält alle notwendigen Definitionen
2
3 %%%%%%%%%%%%%% General
4
5 % Setzt den Professor
6 \def\PROFESSOR{Dr. Supermann}
7 % Setzt den Übungsleiter
8 \def\UEBUNGSLEITER{Someone Günther}
9 % Setzt den Tutor
10 \def\TUTOR{Unbekannt}
11
12 % Setzt den Titel
13 \def\TITLE{Analysis 1 für Inf. und Ing.}
14 % Setzt den Untertitel (Standard)
15 \def\SUBTITLE{\PROFESSOR}
```

```

16 %% Setzt den Titel für ein Übungsblatt (Standard)
17 \def\UEBUNGSHEDER{\TITLE\\Übungsblatt \LILLY@n }
18
19 %% Setzt den Titel für die Titelseite (Standard)
20 \def\FULLTITLE{\TITLE\\\fontsize{18pt}{16pt}\selectfont{\SUBTITLE} }
21
22 %% Setzt den Namen der Vorlesung als Text, siehe \anaI
23 \def\VORLESUNG{\anaI}
24
25 %% Setzt das Intro für eine Zusammenfassung
26 \DeclareRobustCommand{\POLITEINTRO}{\setcounter{TOPICS}{-1}%
27   \TOP[disc]{Disclaimer}{Worte des Autors}
28   Offensichtlich erhebt %... Hier gekürzt
29   \end{center}}
30 }
31
32 %%%%%%%% Layout Control
33
34 %% Setze den Abstand für das Titelbild
35 \providedef{\LILLYxColorxTITLExOffset}{10.3cm}
36 %% Der Zähler für Definitionen wird mit jeder Sektion zurück gesetzt
37 \providedef{\LILLYxBOXxDefinitionxLock}{section}
38 %% Der Zähler für jeden Satz wird mit jeder Sektion zurück gesetzt
39 \providedef{\LILLYxBOXxSatzxLock}{section}
40 %% Bemerkungen sollen ohne Box angezeigt werden
41 \providedef{\LILLYxBOXxBemerkungxBox}{FALSE}
42 %% Beispiele sollen ohne Box angezeigt werden
43 \providedef{\LILLYxBOXxBeispielxBox}{FALSE}
44 %% Beweise sollen ohne Box angezeigt werden
45 \providedef{\LILLYxBOXxBeweisxBox}{FALSE}
46 %% Auf Übungsblättern soll kein Tutorheader angezeigt werden
47 %% (automatisches \TUTORBOX) Dies empfiehlt sich für
48 %% Übungsblätter mit Onlineabgabe, wenn sie gedruckt werden
49 %% sollen, dann: TRUE
50 \providedef{\LILLYxUBxSHOWTUTOR}{FALSE}
51
52 %%%%%%%% Title Control
53
54 %% Setze das Fakultätssymbol
55 \def\LILLYxFACULTY{\LILLYxFACULTYxMATHE}
56 %% Setze die Fakultätsfarbe
57 \def\LILLYxFACULTYxCOLOR{FacultyMathexColor}
58
59 %% Komplett optional, wurde für angenehmere Abstände hier eingefügt
60 \setlength{\itemsep}{0.40\baselineskip}

```

Das ist bezüglich der Konfigurationsdatei eigentlich auch schon alles was gemacht werden muss. Jetzt gilt es allerdings noch das dazugehörige Titelbild zu generieren:

Ein Titelbild erstellen

An sich gibt es keine klare Regel, nach der ein Titelbild generiert werden muss. Es gibt lediglich die folgenden Anforderungen an die sie sich halten sollen:

- ◊ Der Name *muss* `titelimage<Vorlesung>.pdf` entsprechen.
- ◊ Ein Titelbild sollte völlig unabhängig von Lilly erstellt werden.
- ◊ Ein `titelbild` muss komplett in L^AT_EX erstellt werden. Externe Grafiken werden nicht gestattet.
- ◊ Die zugehörige `.tex`-Datei *muss* mitgeliefert werden.
- ◊ Es muss ein *Makefile* existieren welches alle Titelbilder eines Ordners auf einmal neu generiert, sofern eine Änderung vorliegt.
- ◊ Derartige Titelbilder dürfen nur im `Graphics`-Ordner eines Semesters liegen.

Übrigens: Der `TitlepageProvider` ist an sich veraltet und wurde komplett von  `LILLYxPHILOSOPHER` abgelöst.

12

AUSSICHT

DAS WUNDER DER SCHÖPF... EVOLUTION ☺

12.1 Todos

Visuals

Es wäre schön (auch auf Basis von tcolorbox) einige Umgebungen zu haben, mit denen sich Grafiken oder Textabschnitte einfach positionieren lassen. So ist es lästig hierfür jedesmal minipages und unsicher hierfür jedesmal floatings zu verwenden.

Fehler

Das Paket sollte Befehle wie \PackageInfo/Error/Warning unterstützen und auch ausgeben - zudem sollte die komplette Dateistruktur robuster werden und auf Fehler reagieren können

Hoverover tooltips

Eine Idee war es bei Hyperlinks Kommentare mithilfe von Tooltips zu realisieren. Somit wäre es möglich auf den meisten Geräten schnell Informationen zu liefern mithilfe von: Ich bin ein toller Hyperlink.

Weitere

Siehe hier für weitere Todos: <https://github.com/EagleoutIce/LILLY/issues>.

12.2 Geplant für **VER 2.1.0**

Persistence

Geplant ist das Persistence-Paket, welches es ermöglicht Befehle wie die von **LILLYxBOXES** automatisch persistiert.

Weg von xSemester

Die Semesterstruktur soll abgeschafft werden, lediglich der Name der Vorlesung soll das entsprechende Ergebnis liefern. Weiter soll das Konzept der Vorlesung erweitert werden um so allgemeine Datenpakete zur Verfügung zu stellen. Diese sollen weiter auch vom Nutzer in seine Nutzerkonfiguration eingegliedert werden. Allgemein soll es möglich sein, mehrere Pfade zum suchen anzugeben.

Optimize microtype

Es sollen verbesserte Einstellungen für das microtype-Paket gefunden werden.

Different Suffix

Es soll für konfigurationsdateien möglich sein, auch auf .conf zu enden.

Make a changelog

Es sollen die wichtigsten Änderungen in einen Changelog aufgenommen und in der Dokumentation angezeigt werden (neuste Befehle etc.) (Verlinke die Versions-Boxen dorthin.)

Jake Projectfiles & Analysis

Neue **Jake** -Module: Zur Verfügung stellen von Projektdateien, die Gesamtoperationen liefern sowie ausbauen von des Anylsis-Moduls. Weiter soll das autocomplete schneller werden und sich **Jake** bei der Installation an einen gewissen Pfad kopieren um Ort-Unabhängig zu sein!

Renormal aufgabe

normalisiere die Parameter für die Aufgabenbox :D

13

ANHANG

VERALTETE DOKUMENTE, ZUSÄTZLICHES, EASTER-EGGS, ...

13.1 Version VER 1.0.7

13.1.1 Installation in Linux

Da LILLY komplett auf einem Linux-Betriebssystem entwickelt wurde, gestaltet sich die Implementierung relativ einfach. Zuerst gilt es einen neuen Ordner zu erstellen:

```
1 mkdir -p "${HOME}/texmf/tex/latex/"
```

In diesen Ordner (wenn nicht sogar bereits existent) kann nun der gesamte Lilly-Ordner verschoben werden (oder mithilfe eines symbolischen Links verknüpft). Als letztes muss man nun noch TeX über das neue Verzeichnis informieren:

```
1 texhash "${HOME}/texmf"
```

Nun gilt es sich den anderen mitgelieferten Dateien zu widmen! Von besonderer Relevanz ist hierbei `lilly_compile.sh`, welches hier ausführlicher beschrieben wird (REMOVED: OLD). Grundlegend generiert es ein Makefile, das dann zum Kompilieren des Dokuments gedacht ist!

Mithilfe von folgendem Befehl wurde das Makefile für diese Dokumentation generiert:

```
1 ./lilly_compile.sh "Lilly-Dokumentation.doc.tex" \
2           -dir="Dokumentation/"
```

Hierbei wird das Makefile gemäß folgenden Regeln erzeugt:

- ◊ Es soll die tex-Datei: „Lilly-Dokumentation.doc.tex“ kompiliert werden.
- ◊ Das ganze soll (relativ zu `lilly_compile.sh`) im Verzeichnis Dokumentation stattfinden - hier wird ebenfalls das Makefile generiert.

VER 1.0.0

Es wird *nicht* auf die Semantik einzelner Befehle eingegangen!
Copy&Paste ist doof, tippen! ;)

Dies sichert uns die Persistenz des Pakets im Falle einer Neuinstallation/Updates von LATEX

Bemerkung 36 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

VER 1.0.2

Es wird mit den Regeln default, all und clean generiert, selbstredend lässt sich dies erweitern

Mit diesem Makefile kann man nun das Dokument generieren lassen. Zu beachten sei hierbei, dass `make` - im Falle der Regel `all` - Regeln parallel ausführen wird!

Diese Dokumentation wurde mit folgendem Befehl erstellt:

```
1 make "BOXMODE=LIMERENCE"
```

Die Anführungszeichen dienen hier und in anderen Codebeispielen lediglich zur Übersicht!

Hierbei lässt sich ebenfalls erkennen wie sich noch mit dem Makefile einzelne Komponenten (wie das verwendete Boxdesign) ändern lassen!

13.1.2 Spezifikation: Plots

Dieser Abschnitt beschreibt die Richtlinien, auf denen Plots in LILLY integriert werden sollen. Es wurden noch keine (TikZ) basierte Plot-Umgebungen in LILLY integriert.

graph-Environment:

Es soll ein graph-Environment existieren, was auf Basis von PGF das Erstellen folgender Grafiken immens vereinfachen soll:

Aktuell	Ergebnis	Wunsch
<pre> 1 \begin{tikzpicture}[scale=0.6] 2 \draw[help lines, color=gray!30, 3 densely dashed] (-2.4,-0.4) grid (2.4,4.9); 4 \draw[->,thick] (-2.5,0) -- (2.5,0)) 5 node[right]{\$x\$}; 6 \draw[->,thick] (0,-0.5) -- (0,5) node[above]{\$y\$}; 7 \draw[scale=1,domain=-2:2, 9 smooth,variable=\x,purple] 10 plot ({\x},{\x*\x}); 11 \end{tikzpicture} </pre>		<pre> 1 \begin{graph}[scale=0.6,domain=-2:2] 2 \plotline[purple]{\x}{\x*\x}; 3 \end{graph} </pre>

Der Befehl `\plotline` soll hierbei nur in der Umgebung verfügbar sein (TODO: gleiches geplant mit PLA etc.).

Positionierung:

Für die Platzierung von Plots wurden 3 valide Positionen vorgesehen: Zentriert, Links (Text auf rechter Seite), Rechts (Text auf linker Seite). Diese Positionierungen können mithilfe von Floats realisiert werden, sollen aber auf jedenfall auch noch einen absoluten Modus zur Verfügung stellen (primär von zentriert analog zu `\[V]`). Zudem soll das `plot`-Environment selbstverständlich auch ohne Positionierung manuell eingebunden werden können!

13.2 Version VER 1.0.9

13.2.1 Installation in Linux

Für Versionen < 1.0.8 klicke hier: [klick mich!](#)

Da LILLY komplett auf einem Linux-Betriebssystem entwickelt wurde, gestaltet sich die Implementierung relativ einfach. Hierzu nutzen wir das Hilfsprogramm *Jake* welches selbst in C++ geschrieben wurde. Im Folgenden sind die Schritte kurz erklärt.

VER 1.0.8

Installation von *Jake*:

Eine ausführliche Erklärung von *Jake* selbst findest sich weiter hinten ([hier](#)) in dieser Dokumentation:

1. Navigiere mit dem Terminal in das Verzeichnis: `Lilly/Jake/jake_source`
2. Führe nun `make` aus um *Jake* zu kompilieren. Es wird vermutlich kurz dauern, aber danach wird dir das Programm `lilly_jake` zur Verfügung stehen.
3. Nun kannst du dein Terminal neu starten und von überall her `lilly_jake install` aufrufen. Dies sollte den Installationsprozess in Gang setzen.

Für ausführliche Informationen zur Installation konultiere bitte die README-Datei in: [../Lilly/Jake/jake_source/README.md](#). Für Informationen zur Nutzung konultiere: [../Lilly/Jake/README.md](#).

Sollte das Ganze fehlerfrei verlaufen sein, dann: Glückwunsch, du hast Lilly erfolgreich installiert! Betrachte im Falle eines Fehlers bitte erst die Readme-Dateien und die bereits beantworteten Fehler auf Github ([Q](#)) bevor du einen neuen Fehler eröffnest oder mir eine Nachricht schreibst ☺.

Erstellen eines Makefiles:

Nun möchtest du natürlich auch ausprobieren ob die Installation funktioniert hat. Hierzu kannst du in das Testverzeichnis navigieren (`Lilly/Jake/tests`). Hier befinden sich eine Menge Dateien die in dieser Dokumentation auch als Beispiele benutzt werden. Du gibst nun folgendes in die Konsole ein:

```
1 lilly_jake test.tex
```

Jake erstellt nun ein entsprechendes Makefile für dich, welches du nun ausführen kannst:

```
1 make
```

Im Standardmäßig konfigurierten Ausgabe-Ordner `test-OUT` befindet sich nun eine entsprechende PDF Datei ☺.

Bemerkung 37 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

13.2.2 Installation in Mac OS

Entspricht, dank *Jake*, der Linux-Installation.

Hierzu nutzen wir das Hilfsprogramm *Jake* welches selbst in C++ geschrieben wurde. Im Folgenden sind die Schritte kurz erklärt.

Installation von *Jake* :

Eine ausführliche Erklärung von *Jake* selbst findest sich weiter hinten (TODO: LINK) in dieser Dokumentation:

1. Navigiere mit dem Terminal in das Verzeichnis: `Lilly/Jake/jake_source`
2. Führe nun `make` aus um *Jake* zu komplizieren. Es wird vermutlich kurz dauern, aber danach wird dir das Programm `lilly_jake` zur Verfügung stehen.
3. Nun kannst du dein Terminal neu starten und von überall her `lilly_jake install` aufrufen. Dies sollte den Installationsprozess in Gang setzen.

Sollte das Ganze fehlerfrei verlaufen sein, dann: Glückwunsch, du hast Lilly erfolgreich installiert! Betrachte im Falle eines Fehlers bitte erst die Readme-Dateien und die bereits beantworteten Fehler auf Github (Q) bevor du einen neuen Fehler eröffnest oder mir eine Nachricht schreibst ☺.

Erstellen eines Makefiles:

Nun möchtest du natürlich auch ausprobieren ob die Installation funktioniert hat. Hierzu kannst du in das Testverzeichnis navigieren (`Lilly/Jake/tests`). Hier befinden sich eine Menge Dateien die in dieser Dokumentation auch als Beispiele benutzt werden. Du gibst nun folgendes in die Konsole ein:

```
1 lilly_jake test.tex
```

Jake erstellt nun ein entsprechendes Makefile für dich, welches du nun ausführen kannst:

```
1 make
```

Im Standardmäßig konfigurierten Ausgabe-Ordner `test-OUT` befindet sich nun eine entsprechende PDF Datei ☺.

Bemerkung 38 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

Für ausführliche Informationen zur Installation konultiere bitte die README-Datei in: `../Lilly/Jake/jake_source/README.md`.
Für Informationen zur Nutzung konultiere: `../Lilly/Jake/README.md`.

KURZÜBERSICHT

Symbolen

\<(v1.0.2)	107
env@<Sprache>(v1.0.9)	37
env@<Sprache>*(v1.0.9)	37
\<name>(v2.0.0)	29
\>(v1.0.2)	107
\@@CreateNewLectureEvent(v2.0.0)	95
\@@CreateOrnamentCommand(v2.0.0)	81
\@@@university@name(v2.0.0)	118

A

\abs(v1.0.9)	9
\ABSTRACT(v1.0.8)	113
env@abstract(v2.0.0)	115
\Acronym(v2.0.0)	79
\adj(v1.0.3)	9
\aLink(v2.0.0)	108
\ampe1G(v1.0.2)	18
\ampe1H(v1.0.2)	18
\ampe1R(v1.0.2)	18
\ampe1Y(v1.0.2)	18
\anaI(v1.0.1)	71
\arccot(v1.0.8)	10
\attribute(v1.0.9)	24
env@aufgabe(v1.0.0)	59
env@aufgaben(v2.0.0)	100
\AUTHOR(v1.0.4)	73
\AUTHORMAIL(v1.0.4)	73

B

\B(v1.0.3)	10
\b<Sprache>(v1.0.9)	36
\bblock(v1.0.1)	11
env@beispiel(v1.0.0)	57
env@beispiel*(v1.0.0)	57
\BEM(v1.0.0)	50
env@bemerkung(v1.0.0)	56
env@bemerkung*(v1.0.0)	56
env@beweis(v1.0.0)	57
env@beweis*(v1.0.0)	58
\bigcircle(v1.0.0)	69
\bigCRing(v1.0.0)	69
\bigRing(v1.0.0)	69
\blagraphdot(v1.0.2)	19
\bluegraphdot(v1.0.2)	19
\BRIEF(v1.0.8)	113
\btextEmblem(v2.0.0)	29

C

\C(v1.0.0)	10
\c<Sprache>(v1.0.9)	36
\case(v1.0.2)	71
\cd(v1.0.9)	69
\ceil(v2.0.0)	13

env@centered(v1.0.2)	11
\cheaderarrow(v2.0.0)	84
\circlAT(v1.0.1)	11
\cleararrow(v2.0.0)	83
\closeritems(v1.0.3)	68
\cmark(v2.0.0)	101
\codeEmblem(v2.0.0)	27
\ColorfulWords(v2.0.0)	80
\colvec(v1.0.0)	68
\constructList(v2.0.0)	66
\containsList(v2.0.0)	66
\crectat(v1.0.0)	17
\crossAT(v1.0.1)	11
\customex(v1.0.9)	107

D

\das(v1.0.3)	8
\daseq(v1.0.3)	8
\dataInput(v1.0.4)	76
\dateBox(v2.0.0)	62
\debugout(v2.0.0)	73
\DEF(v1.0.0)	50
\default(v1.0.2)	71
env@defaultlst(v2.0.0)	44
env@definition(v1.0.0)	56
env@definition*(v1.0.0)	56
\delete<listName>(v2.0.0)	66
\det(v1.0.3)	9
\diag(v1.0.3)	9
\dif(v2.0.0)	10
\Dim(v1.0.3)	9
\dint(v2.0.0)	10
env@directory(v2.0.0)	24
\dispnote(v1.0.8)	70
env@ditemize(v1.0.4)	99
\dom(v1.0.8)	10
\doublealph(v2.0.0)	80
\DrawTimeTable(v2.0.0)	93

E

env@egraph(v2.0.0)	15
\eidi(v1.0.0)	71
\Eig(v1.0.3)	9
\eig(v1.0.3)	9
\elabel(v2.0.0)	101
\elable(v1.0.0)	101
\emptyset(v1.0.2)	9
\engl(v1.0.8)	69
\entity(v1.0.9)	23
\enum(v1.0.0)	11
env@enumeratea(v1.0.0)	99
\epsilonpsilon(v1.0.3)	10
\errorEmblem(v2.0.0)	27
\etabadd(v2.0.0)	84
\eXButton(v1.0.2)	101

F

\F(v1.0.3)	10
\false(v1.0.0)	73
\fg(v1.0.0)	71
\firstcircle(v1.0.0)	69
\floor(v2.0.0)	13
\folge(v1.0.7)	12
\footnotesizerow(v2.0.0)	84
\fquad(v2.0.0)	69
\fquad(v2.0.0)	69
\fullouterjoin(v2.0.0)	11
\FULLTITLE(v1.0.0)	135

G

\gdb(v1.0.1)	71
\gdra(v1.0.0)	71
\gdw(v1.0.7)	12
\get<ListName>(v2.0.0)	66
\getGraphics(v2.0.0)	25
\getGraphicsPath(v2.0.0)	26
\getPrerendered(v2.0.0)	26
\getSemester(v1.0.4)	76
\git(v1.0.0)	41
\gitRAW(v1.0.0)	41
\newcommand(v1.0.4)	64
\graphdot(v1.0.2)	19
\grad(v1.0.8)	10
env@graph(v1.0.8)	13
\graphdot(v1.0.2)	18
\graphPOI(v1.0.4)	19
\gregraphdot(v1.0.2)	19
\grenewcommand(v1.0.4)	64

H

\HBCColor(v1.0.9)	32
\hcolor(v1.0.9)	32
\headerarrow(v2.0.0)	83
\heute(v2.0.0)	103
\hmark(v1.0.0)	101

I

\i(v1.0.1)	10
\i<Sprache>(v1.0.9)	36
\idxtitle(v2.0.0)	114
\Im(v1.0.2)	9
\imgplaceholder(v1.0.1)	27
\imp(v1.0.0)	107
\include(v1.0.5)	75
\inf(v1.0.6)	9
\infoEmblem(v2.0.0)	27
\infot(v2.0.0)	108
\inputUB(v1.0.3)	50
\inputUBS(v1.0.3)	50
\intro(v1.0.8)	113
\isLanguageLoaded(v2.0.0)	38
\isLanguageNameLoaded(v2.0.0)	38
\isRuntimeLoaded(v2.0.0)	46
\iter<ListName>(v2.0.0)	66

J

\jmark(v1.0.0)	101
\join(v2.0.0)	11

K

\K(v1.0.3)	10
\kattribute(v1.0.0)	24
\KER(v1.0.3)	9
\knn(v1.0.1)	71
\kw(v1.0.2)	107

L

\la(v1.0.0)	71
\leftouterjoin(v2.0.0)	11
env@lemma(v1.0.0)	58
env@lemma*(v1.0.0)	58
\len<ListName>(v2.0.0)	66
\LH(v1.0.3)	9
\lilly@format@iter(v2.0.0)	79
\LILLY@Typ@Dokumentation(v1.0.0)	102
\LILLY@Typ@Mitschrieb(v1.0.0)	102
\LILLY@Typ@Uebungsblatt(v1.0.0)	102
\LILLY@Typ@Zusammenfassung(v1.0.0)	102
\LILLYcoloredSQ(v1.0.6)	70
\LILLYcommand(v1.0.0)	64
\LillyLogo(v2.0.0)	71
\LillyNewLstEnvironCore(v2.0.0)	43
\LillyNewLstEnvironPlain(v2.0.0)	43
\LillyNewLstEnvironPresent(v2.0.0)	44
\lillyPathColorExtension(v1.0.4)	76
\lillyPathConfig(v1.0.4)	74
\lillyPathData(v1.0.4)	74
\lillyPathLayout(v2.0.0)	74
\LILLYxABSPATHx<Module>(v1.0.4)	76
\LILLYxABSPATHxBEAMER(v1.0.4)	76
\LILLYxABSPATHxCONTROLLERS(v1.0.4)	76
\LILLYxABSPATHxCORE(v1.0.4)	76
\LILLYxABSPATHxDATA(v1.0.4)	76
\LILLYxABSPATHxFALLBACKS(v1.0.4)	76
\LILLYxABSPATHxGRAPHICS(v1.0.4)	76
\LILLYxABSPATHxHELPER(v1.0.4)	76
\LILLYxABSPATHxLISTINGS(v1.0.4)	76
\LILLYxABSPATHxMATHS(v1.0.4)	76
\LILLYxABSPATHxPRESENTER(v1.0.4)	76
\LILLYxABSPATHxUTIL(v1.0.4)	76
\LILLYxBOXx<Bezeichner>xBox(v1.0.8)	48
\LILLYxBOXx<Bezeichner>xEnable(v1.0.8)	48
\LILLYxBOXx<Bezeichner>xLock(v1.0.8)	48
\lillyxBOXx<BoxID>xBoxCol(v2.0.0)	52
\lillyxBOXx<BoxID>xBoxEnabled(v2.0.0)	52
\lillyxBOXx<BoxID>xCreateList(v2.0.0)	52
\lillyxBOXx<BoxID>xCustomList(v2.0.0)	52
\lillyxBOXx<BoxID>xEmblem(v2.0.0)	52
\lillyxBOXx<BoxID>xInCode(v2.0.0)	52
\lillyxBOXx<BoxID>xListMen(v2.0.0)	52
\lillyxBOXx<BoxID>xListName(v2.0.0)	52
\lillyxBOXx<BoxID>xListText(v2.0.0)	52
\lillyxBOXx<BoxID>xLock(v2.0.0)	52
\lillyxBOXx<BoxID>xName(v2.0.0)	52
\lillyxBOXx<BoxID>xOutCode(v2.0.0)	52
\lillyxBOXx<BoxID>xPostCode(v2.0.0)	52
\lillyxBOXx<BoxID>xPreCode(v2.0.0)	52
\lillyxBOXx<BoxID>xTitle(v2.0.0)	52
\lillyxBOXx<BoxID>xUseBox(v2.0.0)	52
\lillyxBOXx<BoxID>xUseStyle(v2.0.0)	52
\LILLYxBOXxHIGHLEVELxLOCK(v1.0.8)	48
\LILLYxBOXxMODE(v1.0.5)	48
\LILLYxCLEARxHEADFOOT(v2.0.0)	103
\LILLYxCLSPATH(v1.0.4)	73
\LILLYxCMD(v1.0.4)	75
\LILLYxColorxInject(v1.0.1)	32

\LILLYxCOLORxRainbow(v1.0.6)	71
\LILLYxColorxTITLExSETTINGSxGENERAL(v1.0.9)	118
\LILLYxColorxTITLExSETTINGSxVORLESUNG(v1.0.9)	118
\LILLYxDEBUG(v2.0.0)	73
\LILLYxDemandPackage(v1.0.7)	75
\LILLYxDOC PATH(v1.0.4)	73
\LILLYxDOCUMENTNAME(v1.0.4)	73
\LILLYxDOCUMENTxSUBNAME(v1.0.5)	75
\LILLYxEXTERNALIZE(v1.0.9)	74
\LILLYxFACULTYxMATHE(v1.0.8)	118
\LILLYxFACULTYxPRAKTIISCHEINFORMATIK(v1.0.8)	118
\LILLYxFACULTYxTECHNISCHEINFORMATIK(v1.0.8)	118
\LILLYxFACULTYxTHEORETISCHEINFORMATIK(v1.0.8)	118
\LILLYxFlavourText(v1.0.4)	74
\LILLYxFooterxBUTTONS(v1.0.0)	102
\LILLYxGENxFACULTY(v1.0.8)	118
\LILLYxHYPERLINK(v1.0.2)	101
\LILLYxIMAGESxShow(v1.0.3)	102
\lillyxINFOBOXESx<InfoBox>xBgColor(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xBorderColor(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xDraw(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xEmblem(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xFgColor(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xInCode(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xMarker(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xOutCode(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xPostCode(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xPreCode(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xTextFont(v2.0.0)	62
\lillyxINFOBOXESx<InfoBox>xTitleCode(v2.0.0)	63
\lillyxINFOBOXESx<InfoBox>xTitleFont(v2.0.0)	62
\LILLYxLISTINGSxFONTSIZE(v2.0.0)	40
\LILLYxLISTINGSxNUMxFONTSIZE(v2.0.0)	40
\LILLYxLoadPackage(v1.0.7)	75
\LILLYxlstTypeWriter(v1.0.0)	40
\LILLYxMathMode(v1.0.3)	8
\LILLYxMODE(v1.0.0)	102
\LILLYxMODExDEFAULT(v1.0.0)	102
\LILLYxMODExDUMMY(v1.0.0)	102
\LILLYxMODExEVENT(v1.0.4)	73
\LILLYxMODExPRINT(v1.0.0)	102
\LILLYxPATHx<Module>(v1.0.4)	76
\LILLYxPATHxBEAMER(v1.0.4)	76
\LILLYxPATHxCONTROLLERS(v1.0.4)	76
\LILLYxPATHxCORE(v1.0.4)	76
\LILLYxPATHxDATA(v1.0.4)	76
\LILLYxPATHxFALLBACKS(v1.0.4)	76
\LILLYxPATHxFILExROOT(v1.0.4)	76
\LILLYxPATHxGRAPHICS(v1.0.4)	76
\LILLYxPATHxHELPER(v1.0.4)	76
\LILLYxPATHxINDEX(v1.0.4)	76
\LILLYxPATHxLISTINGS(v1.0.4)	76
\LILLYxPATHxMATHS(v1.0.4)	76
\LILLYxPATHxPRESENTER(v1.0.4)	76
\LILLYxPATHxROOT(v1.0.4)	76
\LILLYxPATHxUTIL(v1.0.4)	76
\LILLYxPHILOSOPHERxBONUSxTTOCxHEADER(v1.0.9)	120
\LILLYxPHILOSOPHERxBORDERBLOCK(v1.0.9)	119
\LILLYxPHILOSOPHERxINIT(v1.0.9)	119
\LILLYxPHILOSOPHERxMETADATA(v1.0.9)	119
\LILLYxPoliteKnock(v1.0.7)	74
\LillyxRestorexCurrentColorProfile(v2.0.0)	32
\LILLYxSemester(v1.0.0)	74
\LILLYxSTATUS(v1.0.0)	72
\LillyxStorexCurrentColorProfile(v2.0.0)	32
\LILLYxTITLE(v1.0.9)	122
\LILLYxTITLExBONUS(v1.0.9)	120
\LILLYxTITLExOffset(v1.0.0)	136

\LILLYxTITLExRAW(v1.0.9)	122
\LILLYxUSure(v1.0.7)	74
\LILLYxVERSION(v1.0.0)	72
\LILLYxVERSIONxLONG(v1.0.0)	72
\LILLYxWANNABExERROR(v1.0.7)	74
\LILLYxwriteLst(v1.0.8)	39
\limk(v1.0.7)	12
\limn(v1.0.7)	12
\linclude(v1.0.5)	75
\linput(v1.0.5)	75
\liste(v1.0.0)	11
\LK(v1.0.3)	9
\LoadLillyBoxMode(v2.0.0)	48
\loopBot(v1.0.3)	21
\loopLeft(v1.0.3)	21
\loopRight(v1.0.3)	21
\loopTo(v1.0.3)	21
\loopTop(v1.0.3)	21
\lpage(v2.0.0)	101
\lreqn(v1.0.0)	68
\lstcomment(v2.0.0)	40
\lstfs(v2.0.0)	68
\lstkwfive(v2.0.0)	41
\lstkwfour(v2.0.0)	41
\lstkwone(v2.0.0)	41
\lstkwsix(v2.0.0)	41
\lstkwtwo(v2.0.0)	41
\lstnumber(v2.0.0)	40
env@ lstplain (v1.0.9)	39
\lstshowcmd(v2.0.0)	39
\lststring(v2.0.0)	40
\ltt(v2.0.0)	68

M

\makeenvglobal(v2.0.0)	65
\makerenewglobal(v2.0.0)	64
\makerenewlocal(v2.0.0)	64
\margraphdot(v1.0.2)	19
\mathEmblem(v2.0.0)	27
env@ matrix (v1.0.2)	9
\max(v1.0.6)	9
\MHeaderRow(v2.0.0)	87
\min(v1.0.6)	9
\minicolvec(v1.0.0)	68
env@ mtable (v2.0.0)	86
env@ mtabular (v2.0.0)	88
\MNHeaderRow(v2.0.0)	87
\MTBHeaderRow(v2.0.0)	88
\mod(v1.0.2)	9
env@ mtable (v2.0.0)	86
env@ mtabular (v2.0.0)	87
\MTBHeaderRow(v2.0.0)	88

N

\N(v1.0.0)	10
\n@false(v1.0.4)	73
\n@true(v1.0.4)	73
\narrowitems(v1.0.0)	68
\negaskip(v1.0.9)	107
\negbskip(v1.0.9)	107
\neuronSquare(v1.0.5)	23
\new<EventID>(v2.0.0)	95
\NewEmblem(v2.0.0)	28
\NewInfoBox(v2.0.0)	60
\NewLectureSeries(v2.0.0)	95

\NewTimeTable(v2.0.0)	91
\NewTimeTableEvent(v2.0.0)	94
\NoFormatChar(v2.0.0)	79
\nographdot(v1.0.2)	19
\normalrow(v2.0.0)	84
\note(v1.0.8)	70
\nskip(v1.0.9)	70
env@nstabbing(v1.0.2)	11

O

\O(v1.0.3)	10
\Obda(v1.0.8)	12
\obda(v1.0.8)	12
\oragraphdot(v1.0.2)	19
\orna<Name>(v2.0.0)	81
\OrnamentsBoxTitle(v2.0.0)	81
\OrnamentsLower(v2.0.0)	81
\OrnamentsUpper(v2.0.0)	81
\overbar(v1.0.3)	8

P

\P(v1.0.3)	10
\p<lang>(v2.0.0)	46
\p<Sprache>(v2.0.0)	36
\pdः(v1.0.1)	71
\PgetX(v2.0.0)	19
\PgetXY(v2.0.0)	19
\PgetY(v2.0.0)	19
\phi(v1.0.3)	10
\PickRandom(v2.0.0)	67
env@plain<Sprache>(v2.0.0)	37
env@plainlist(v2.0.0)	44
\plotline(v1.0.8)	13
\plotseq(v1.0.8)	13
env@pmatrix(v1.0.0)	9
\pnpsetauthor(v2.0.0)	114
\pnpsubtitle(v2.0.0)	114
\pnpsettle(v2.0.0)	114
\pnptitle(v2.0.0)	114
\pnptoc(v2.0.0)	114
env@poem(v2.0.0)	89
env@poem*(v2.0.0)	89
\poemssetauthor(v2.0.0)	89
\poemssetsubtitle(v2.0.0)	115
\poemssettitle(v2.0.0)	115
\poemssettocpoemsheader(v2.0.0)	115
\poemssettocquotesheader(v2.0.0)	115
\poemstitle(v2.0.0)	115
\poemstoc(v2.0.0)	115
\points(v1.0.6)	105
\POLITEINTRO(v1.0.0)	136
\PoliteWords(v2.0.0)	80
\POLYRAD(v1.0.2)	18
\present<Name>(v2.0.0)	93
\PresentAllOrnaments(v2.0.0)	82
env@present1st(v2.0.0)	44
\PresentTimeTable(v2.0.0)	93
\printbib(v2.0.0)	104
\printHeader(v1.0.8)	113
\printLILLY(v1.0.9)	113
\printMiniToc(v2.0.0)	111
\PROFESSOR(v1.0.0)	135
\providedef(v2.0.0)	65
\pурgraphdot(v1.0.2)	19
\pusList(v2.0.0)	67
\pvs(v1.0.1)	71

Q

\Q(v1.0.0)	10
\q(v1.0.1)	68
\qedsymbol(v1.0.0)	69
\quesad(v1.0.3)	8
\qq(v1.0.1)	68
env@quote(v2.0.0)	91
env@quotes(v2.0.0)	90
env@quotes*(v2.0.0)	91

R

\R(v1.0.0)	10
\r<Runtime>(v2.0.0)	46
\RandomFlavourText(v2.0.0)	117
\RandomInt(v2.0.0)	67
\RawTimeTableEvent(v2.0.0)	93
\Re(v1.0.2)	9
\rectat(v1.0.0)	17
\reg(v1.0.2)	107
\RegisterBox(v2.0.0)	51
\registerColors(v2.0.0)	30
\RegisterLanguage(v2.0.0)	42
\reihe(v1.0.7)	12
\relation(v1.0.9)	24
\rg(v1.0.3)	9
\rightouterjoin(v2.0.0)	11
\ring(v1.0.0)	69
\ringC(v1.0.0)	69
\rom(v1.0.0)	69
\rotateRPY(v1.0.4)	20

S

env@s<Sprache>(v2.0.0)	38
\sad(v1.0.3)	8
env@satz(v1.0.0)	57
env@satz*(v1.0.0)	57
\say(v1.0.0)	69
\sch(v1.0.8)	10
\scriptsizerow(v2.0.0)	84
\secondcircle(v1.0.0)	69
\sel(v1.0.3)	9
\setLillyAuthor(v2.0.0)	71
\setLillyAuthormail(v2.0.0)	71
\setLinkColor(v2.0.0)	101
\setList(v2.0.0)	67
\SetPartFlavour(v2.0.0)	111
\setrow(v2.0.0)	83
\shouldeq(v1.0.6)	8
\showcase(v2.0.0)	108
\sign(v1.0.3)	9
\silentHmark(v1.0.9)	101
\singlequote(v2.0.0)	91
env@smalldesc(v2.0.0)	108
env@smalldite(v1.0.0)	108
\smallrow(v2.0.0)	84
\snote(v1.0.8)	70
env@sqcases(v1.0.2)	11
\sqrt(v1.0.3)	9
\sr(v1.0.2)	107
\startAppendix(v1.0.2)	107
\startAppendix(v1.0.8)	113
\store<ListName>(v2.0.0)	66
\subduelines(v2.0.0)	89
\SUBTITLE(v1.0.0)	135
\sumk(v1.0.7)	12

\sumn ^(v1.0.7)	12
\sup ^(v1.0.6)	9
\sw ^(v1.0.2)	107
env@switch ^(v1.0.2)	71
T	
\T ^(v1.0.0)	68
\tab ^(v1.0.0)	68
\tabadd ^(v2.0.0)	84
\tabforeach ^(v2.0.0)	84
\TableOfContents ^(v2.0.0)	111
\tabprint ^(v2.0.0)	84
\tabreset ^(v2.0.0)	84
env@task ^(v1.0.0)	50
\textEmblem ^(v2.0.0)	29
\tgraphdot ^(v1.0.2)	18
\the<ListName> ^(v2.0.0)	66
\the<Name> ^(v2.0.0)	93
\thirdcircle ^(v1.0.0)	69
env@tikzternal ^(v1.0.7)	27
\tinyrow ^(v2.0.0)	84
\TITLE ^(v1.0.0)	135
\TITLE ^(v1.0.8)	113
\TitleSUB ^(v2.0.0)	104
\TOP ^(v1.0.3)	107
\TOPskip ^(v2.0.0)	107
\TransformBox ^(v2.0.0)	52
\trenner ^(v1.0.0)	12
\true ^(v1.0.0)	73
\TUTOR ^(v1.0.0)	135
\TUTORBOX ^(v1.0.1)	105
\typesetList ^(v2.0.0)	66
U	
env@uebungsbatt ^(v1.0.0)	59

env@uebungsbatt* ^(v1.0.1)	59
\UEBUNGSHADER ^(v1.0.0)	135
\UEBUNGSLEITER ^(v1.0.0)	135
\updateColors ^(v2.0.0)	30
\userput ^(v1.0.4)	76

V

\val ^(v1.0.8)	10
\VORLESUNG ^(v1.0.0)	135
\VRule ^(v1.0.4)	12

W

\warningEmblem ^(v2.0.0)	27
env@wgraph ^(v1.0.8)	15

X

\X ^(v1.0.3)	10
\x ^(v1.0.2)	13
\xa ^(v1.0.1)	11
\xb ^(v1.0.1)	11
\xc ^(v1.0.1)	11
\xmark ^(v2.0.0)	13

Y

\y ^(v1.0.2)	12
\ymark ^(v2.0.0)	13

Z

\z ^(v1.0.0)	10
\z ^(v1.0.2)	12
env@zusammenfassung ^(v1.0.0)	58
env@zusammenfassung* ^(v1.0.0)	58