Beschreibung der LaTeX-Pakete

sets und vhistory

Jochen Wertenauer

jwertenauer@gmx.de

22. Januar 2013

Inhaltsverzeichnis

1 Einleitung

Es ist mir klar, dass die wenigsten Leser Einleitungen lesen. Ich empfehle dennoch, diesen Abschnitt nicht zu überspringen, da er zu erläutern versucht, warum die Pakete sets und vhistory entwickelt wurden. So können Sie frühzeitig erkennen, ob sie Ihren Anforderungen gerecht werden.

Bei Softwareprojekten entstehen (hoffentlich) viele Dokumente wie Spezifikation oder Entwurf. Diese Dokumente werden mehrfach überarbeitet. Um Änderungen direkt nachvollziehen zu können, sollten diese Dokumente eine sogenannte *Versionshistorie* enthalten. Dabei handelt es sich um eine Tabelle, deren Einträge folgende Daten umfassen:

- eine Versionsnummer,
- das Datum der Änderung,
- die Kürzel der Personen, die die Änderungen vorgenommen haben (die Autoren),
- eine Beschreibung der Änderungen.

Bestimmte Daten der Versionshistorie sollen häufig an anderen Stellen im Dokument wiederholt werden. So soll typischerweise die Titelseite die aktuelle Versionsnummer und alle Autoren aufführen. Die Versionsnummer sollte außerdem auf allen Seiten des Dokuments, z. B. in einer Fußzeile, wiederholt werden. Dadurch kann leicht überprüft werden, ob eine Seite zur aktuellsten Version gehört oder schon veraltet ist.

Normalerweise werden die Daten, die z. B. auf der Titelseite erscheinen, nicht aus der Versionshistorie übernommen, sondern an anderer Stelle erneut angegeben. Die Einträge der Versionshistorie werden in der Regel immer aktualisiert. In der Hektik wird aber meist vergessen, die Angaben für Titelseite etc. zu aktualisieren. Das Ergebnis sind inkonsistente Dokumente. Aus eigener Erfahrung weiß ich, dass die Angaben zu den Autoren praktisch nie stimmen, besonders wenn im Laufe der Zeit mehrere Personen an einem Dokument gearbeitet haben.

Es wäre also schön, wenn der Autor eines Dokuments sich nur darum kümmern müsste, die Versionshistorie auf dem aktuellen Stand zu halten. Die Informationen auf der Titelseite und in Fußzeilen sollten automatisch aus der Versionshistorie generiert werden.

Diese Anforderungen sind ohne einen gewissen Aufwand nicht umzusetzen, da beispielsweise die Titelseite erzeugt wird, bevor die Versionshistorie überhaupt gelesen wurde. Die relevanten Daten müssen deshalb in eine Datei geschrieben und noch vor Bearbeitung der Titelseite wieder eingelesen werden.

Da für manche Anwendungen auch der Zeitpunkt, zu dem die aux-Datei eingelesen wird zu spät ist, wird eine eigene Datei mit der Endung hst angelegt. Für die Tabelle mit der Versionshistorie ist ebenfalls eine eigene Datei notwendig, doch dazu später.

Ein anderes Problem stellt die Liste der Autoren dar. Diese Liste kann nicht einfach durch Aneinanderreihung der Autor-Einträge in der Versionshistorie erzeugt werden, da sonst einige Personen mehrfach auftreten würden. Dies war die Geburtsstunde des Pakets sets, mit dem einfache Mengen von Text verwaltet werden können. Die Menge aller Autoren wird bei jedem Eintrag in der Versionshistorie mit der Menge der angegebenen Autoren vereinigt. Die Menge aller Autoren kann dann in alphabetisch sortierter Form an beliebiger Stelle – eben meist auf der Titelseite – ausgegeben werden.

Soweit zur Vorrede. Die beiden folgenden Abschnitte beschreiben die beiden Pakete eingehender und zeigen, wie man mit ihnen arbeitet. Dabei wurde darauf verzichtet, den Quellcode der Pakete wiederzugeben. Wer sich dafür interessiert, kann direkt in die Quellen schauen. Ich habe versucht, den Quellcode so zu strukturieren und zu kommentieren, dass er lesbar ist.

2 Das Paket vhistory

Sinn und Zweck dieses Pakets wurde bereits ausführlich in Abschnitt 1 beschrieben. Ich sagte ja, dass es sich lohnt, die Einleitung zu lesen. Hier soll nun ausführlich erklärt werden, wie man mit vhistory arbeitet.

2.1 Laden des Pakets

Das Paket wird wie üblich in der Präambel mit dem Befehl

```
\usepackage{vhistory}
```

geladen. Vhistory setzt LaTeX 2ε und die Pakete sets und ltxtable (welches wiederum longtable und tabularx benötigt) voraus. Sollten die Pakete noch nicht geladen worden sein, werden sie von vhistory automatisch geladen.

Das Paket vhistory versteht einige Optionen, um sein Verhalten anzupassen. Diese sind im Folgenden aufgelistet. Ein Aufruf mit Optionen lautet zum Beispiel

```
\usepackage[tocentry, owncaptions]{vhistory}.
```

nochapter: Ist diese Option beim Laden des Pakets angegeben worden, wird für die Versionshistorie kein eigenes Kapitel – bzw. kein eigener Abschnitt, falls die Dokumentenklasse article (oder scrartcl) verwendet wird – erzeugt.

tocentry: Mit dieser Option wird veranlasst, dass die Versionshistorie im Inhaltsverzeichnis aufgeführt wird. Normalerweise wird dieser Eintrag nicht erzeugt. Ist die Option nochapter aktiviert hat tocentry keine Funktion. Hier sind Sie selbst für eventuelle Einträge in das Inhaltsverzeichnis verantwortlich.

owncaptions: vhistory unterstützt die Sprachen Deutsch, Englisch, Französisch, Kroatisch und Niederländisch. Verwenden Sie eine nicht unterstützte Sprache, werden die Überschriften (z. B. „Versionshistorie“ oder „Änderungen“) in Englisch ausgegeben. In diesem Fall möchten Sie vielleicht die Überschriften selbst verändern. Die Option owncaptions unterstützt Sie dabei. Näheres zu diesem Thema finden Sie in Unterabschnitt 2.3.

tablegrid: Mit dieser Option wird in der Tabelle mit der Versionshistorie ein Gitternetz angezeigt.

2.2 Verwendung

Die Verwendung von vhistory ist denkbar einfach und soll in diesem Unterabschnitt beschrieben werden. Allgemein gilt, dass vhistory zwei Durchläufe benötigt, da Daten in Dateien geschrieben werden.

2.2.1 Erzeugen der Versionshistorie

Die Versionshistorie wird als Umgebung dargestellt:

```
\begin{versionhistory}
<Einträge>
\end{versionhistory}
```

Ein Eintrag hat die allgemeine Gestalt:

```
\vhEntry{<Version>}{<Datum>}{<Autoren>}{<Änderungen>}
```

Die Autoren werden in der Mengenschreibweise des Pakets set angegeben, d. h. als Trennzeichen wird `|` verwendet. Ein Eintrag könnte also wie folgt aussehen:

```
\vhEntry{1.1}{13.05.04}{JW|AK|KL}{Fehler korrigiert.}
```

Durch das `\begin...` wird ein neues Kapitel (beziehungsweise ein neuer Abschnitt, falls article verwendet wird) begonnen, wenn dies nicht durch die Paketoption nochapter wie oben beschrieben abgeschalten wurde.

Die Versionshistorie selbst wird in eine „ltxtable“ gesetzt. Dadurch kann die Versionshistorie auch mehrere Seiten umfassen. Die Spalten „Autor(en)“ und „Änderungen“ werden automatisch umgebrochen. Das Paket ltxtable setzt voraus, dass die Tabelle in einer eigenen Datei liegt. Diese Datei wird von vhistory automatisch erzeugt und hat die Endung „ver“.

2.2.2 Auslesen der aktuellen Versionsnummer

Die aktuelle Versionsnummer – genauer: die zuletzt angegebene Versionsnummer – kann mit dem Befehl

```
\vhCurrentVersion
```

bestimmt werden. Der Befehl ist ab der Einbindung des Pakets verfügbar.

2.2.3 Auslesen des aktuellen Datums

Analog zur aktuellen Versionsnummer kann auch das Datum der letzten Änderung angezeigt werden. Dies funktioniert mit dem Befehl

```
\vhCurrentDate.
```

Der Befehl ist ebenfalls ab der Einbindung des Pakets verfügbar.

2.2.4 Liste aller Autoren ausgeben

An eine Liste der Autoren kommen Sie auf zwei Wege. Über das Kommando

`\vhAllAuthorsSet`

können Sie sich die Autoren als Menge wie in Abschnitt 3 beschrieben zurückgeben lassen. Der weitaus einfachere Weg liegt im Kommando

`\vhListAllAuthors`.

Dieses Kommando gibt eine alphabetisch sortierte Liste der Autoren aus. Die einzelnen Einträge werden durch Kommata getrennt. Soll stattdessen ein anderes Trennzeichen – beispielsweise `&` zur Ausgabe in einer Tabelle – verwendet werden, können sie den Befehl

`\setseparator`

aus dem Paket sets umdefinieren (siehe auch Abschnitt 3.1.2).

Manchmal möchten Sie vielleicht, dass die Autorenliste komplette Namen statt Kürzeln enthält, zum Beispiel „Jochen Wertenauer“ statt des Kürzels „JW“, das in der Versionshistorie Verwendung findet. Indem Sie das Kommando

`\vhListAllAuthorsLong`

verwenden, erhalten Sei das gewünschte Verhalten. In diesem Fall schreiben Sie weiterhin “JW” im `\entry`-Kommando (Hinweis: Da ist kein backslash an dieser Stelle!), definieren aber zusätzlich das Makro `\JW` wie unten beschrieben.

```
\newcommand{\JW}{Jochen Wertenauer}
```

In der Versionshistorie wird weiterhin der Text „JW“ angezeigt, aber das Kommando `\vhListAllAuthorsLong` verwendet das Makro `\JW`. Ist das Makro undefiniert, wird das Kürzel ausgegeben.

Alternativ zu `\vhListAllAuthorsLong` können Sie auch

`\vhListAllAuthorsLongWithAbbrev`

verwenden, das hinter dem kompletten Namen jeweils das Kürzel in Klammern ausgibt, also z. B. „Jochen Wertenauer (JW)“. Name und Kürzel werden durch den Befehl

`\vhAbbrevSeparator`

getrennt. Dieser ist mit „`\␣`“ vorbelegt. Falls Sie Zeilenumbrüche vermeiden wollen, können Sie den Befehl auch mit

```
\renewcommand{\vhAbbrevSeparator}{~}
```

umdefinieren. Die Klammern können durch Umdefinition von

`\vhAbbrevLeft` und

`\vhAbbrevRight`

ebenfalls den eigenen Vorlieben angepasst werden.

	\vhhistoryname	\vhversionname	\vhdatename	\vhauthorname	\vhchangename
Deutsch	Versionshistorie	Version	Datum	Autor(en)	Änderungen
Englisch	Revision History	Revision	Date	Author(s)	Description
Französisch	Historique	Version	Date	Auteur(s)	Modifications
Kroatisch	Povijest verzija	Verzija	Datum	Autor(ica)	Opis Promjena
Niederländisch	Wijzigingen	Herziening	Datum	Auteur(s)	Beschrijving

Tabelle 1: sprachabhängige Texte

2.2.5 Spaltenbreite der Versionshistorie anpassen

Sollte die Breite der Spalten für Autoren und Änderungen in der Versionshistorie nicht Ihren Bedürfnissen entsprechen, können Sie das Breitenverhältnis der beiden Spalten zueinander anpassen. Die Befehlsfolge

```
\renewcommand \vhAuthorColWidth{.8\hsize}
\renewcommand \vhChangeColWidth{1.2\hsize}
```

ändert das Verhältnis auf 2:3 (Standard ist 1:3). Bitte achten Sie bei Änderungen darauf, dass die Summe der beiden Breiten `2\hsize` ergibt.

2.3 Sprachunterstützung

Wie schon in Abschnitt 2.1 erwähnt unterstützt vhistory die Sprachen Deutsch, Englisch, Französisch und Niederländisch. Die sprachabhängigen Texte mit ihren Voreinstellungen sind in Tabelle 1 aufgelistet. Soll das Dokument in einer Sprache verfasst werden, die vhistory nicht unterstützt, wird die englische Variante gewählt.

Mit der Paketoption „owncaptions“ können eigene Überschriften verwendet werden. Die Option veranlasst vhistory dazu, die Kommandos, die die Überschriften enthalten, mit den Varianten der aktuell gewählten Sprache vorzubelegen. Deshalb ist es sinnvoll, in diesem Fall Pakete wie babel oder ngerman vor vhistory zu laden.

Über das Kommando

```
\renewcommand{<Kommando>}{<gewünschter Text>}
```

kann eine Überschrift verändert werden. Wollen Sie beispielsweise statt „Änderungen“ lieber „Verbesserungen“ verwenden, schreiben Sie

```
\renewcommand{\vhchangename}{Verbesserungen}.
```

2.4 Beispiel

```
1:  \documentclass{scrartcl}
2:  \usepackage{ngerman, vhistory, hyperref}
3:
4:  \newcommand{\docTitle}{Ein Beispiel f\"ur vhistory}
5:
6:  \hypersetup{%
7:    pdftitle  = {\docTitle},
8:    pdfkeywords = {\docTitle, Version \vhCurrentVersion
9:                   vom \vhCurrentDate},
10:   pdfauthor = {\vhAllAuthorsSet}
11: }
12:
13: \usepackage{scrpage2}
14: \pagestyle{scrheadings}
15: \ifoot{\docTitle\ -- Version \vhCurrentVersion}
16: \cfoot[]{}
17: \ofoot[\thepage]{\thepage}
18:
19: \begin{document}
20:
21: \title{\docTitle}
22: \author{\vhListAllAuthors}
23: \date{Version \vhCurrentVersion\ vom \vhCurrentDate}
24: \maketitle
25:
26: \begin{versionhistory}
27:   \vhEntry{1.0}{22.01.04}{JPW|KW}{Erstellung}
28:   \vhEntry{1.1}{23.01.04}{DP|JPW}{Fehlerkorrektur}
29:   \vhEntry{1.2}{03.02.04}{DP|JPW}{\"Uberarbeitung nach Review}
30: \end{versionhistory}
31:
32: \end{document}
```

Abbildung 1: Beispiel für die Verwendung von vhistory

3 Das Paket sets

Wie schon in der Einleitung beschrieben ist sets dazu konzipiert, Mengenoperationen zu unterstützen. Die Elemente einer Menge sind normalerweise einfacher Text, Sie können aber auch Kommandos in Mengen einfügen. Diese werden – außer bei der Ausgabe – nicht ausgepackt. Die Verwendung von geschweiften Klammern in Mengen funktioniert leider nicht. In diesem Fall müssen Sie sich eine Abkürzung definieren, die ohne Parameter auskommt. Parameter ohne Klammern funktionieren jedoch. „`H"agar`“ wäre also ein gültiges Element einer Menge. Die Elemente „`\endset`“ und „`\empty`“ dürfen nicht in einer Menge enthalten sein.

Da ein Dokument nur wenige Autoren hat, wurde auf Effizienz kein besonderer Wert gelegt. Die Mengen sollten deshalb relativ klein sein. Sollten Sie dennoch eine Menge mit hunderten oder gar tausenden von Elementen anlegen wollen, kann es passieren, dass der Stack von TEX überläuft.

Normalerweise ist bei einer Menge die Reihenfolge der Elemente egal. Dies ist auch hier bei den meisten Befehlen der Fall. Bei Abweichungen wird darauf hingewiesen.

Das Paket sets benötigt LATEX 2_ε.

3.1 Verwendung

In diesem Unterabschnitt soll die Verwendung des Pakets sets vorgestellt werden. Dabei werden einige Beispielmengen verwendet werden:

$$\begin{aligned} A &= \{Alice, Bob, Charly\} \\ B &= \{Alice, Bob\} \\ C &= \{Bob, Dean\} \\ D &= \{Dean\} \\ L &= \emptyset \end{aligned}$$

3.1.1 Konstruktoren

Um eine Menge anzulegen, gibt es die Befehle

```
\newset{<Menge>}{<Inhalt>}
```

und

```
\newsetsimple{<Menge>}{<Inhalt>}.
```

`<Menge>` ist ein Kommandoname, unter dem die Menge später erreichbar sein soll. Die Elemente einer Menge werden durch `|` getrennt. Die Menge A ließe

sich also wie folgt definieren:

```
\newset{\mA}{Alice|Bob|Charly}
```

Die Menge L wird mit

```
\newset{\mL}{}
```

definiert.

`\newset` legt also eine neue Menge an. Diese wird dabei alphabetisch sortiert und Duplikate werden entfernt. Es wäre also egal gewesen, wenn bei der Definition von A nach „Charly“ ein weiteres Mal „Alice“ gestanden hätte.

Der Aufwand für Sortierung und Duplikatentfernung ist an dieser Stelle unnötig. Will man diese Schritte nicht durchführen lassen, kann man eine Menge auch mit `\newsetsimple` definieren.

Da sie später noch benötigt werden, werden wir nun alle oben genannten Mengen anlegen:

```
\newsetsimple{\mA}{Alice|Bob|Charly}
\newsetsimple{\mD}{Alice|Bob}
\newsetsimple{\mC}{Bob|Dean}
\newsetsimple{\mD}{Dean}
\newsetsimple{\mL}{}
```

3.1.2 Inspektoren

Inspektoren dienen dazu, Eigenschaften von Mengen herauszufinden oder diese auszugeben.

Ausgabe: Eine Menge lässt sich über

```
\listset
```

ausgeben. Die Elemente werden in der Reihenfolge, in der sie in der Menge stehen, ausgegeben. Als Trennzeichen zwischen den Elementen wird das Komma verwendet.

`\listset{\mA}` ergibt also folgende Ausgabe:

Alice, Bob, Charly

Manchmal möchte man die Elemente einer Menge vielleicht auf andere Art und Weise trennen, z. B. mit einem `&` zur Darstellung in einer Tabelle. Hier hilft einem die (temporäre) Umdefinition des Kommandos

```
\setseparator
```

weiter. Normalerweise hat dieses Makro den Ersetzungstext ‘, \␣’.

Größenbestimmung: Der nächste Inspektor hat die Syntax

```
\sizeofset{M}\is{<Zähler>}
```

<Zähler> ist dabei der Name eines LaTeX-Zählers, in dem die Anzahl der Elemente der Menge M gespeichert wird. Die Kommandosequenz

```
\newcounter{mycounter}
\sizeofset{\mB}\is{mycounter}
\arabic{mycounter}
```

führt zur Ausgabe: „2“ Wird die Größe der Menge L bestimmt, ist das Ergebnis wie erwartet „0“.

Prüfung auf Mitgliedschaft: Mit dem Befehl

```
\iselementofset{e}{M}
```

kann überprüft werden, ob $e \in M$ gilt. Der Aufwand ist $O(1)$, da alle Arbeit durch die Mustererkennung von TeX übernommen wird. Die Befehlssequenz

```
\if \iselementofset{Bob}{\mC}Ja\else Nein\fi
```

würde zur Ausgabe „Ja“ führen, der gleiche Test mit Menge D zu „Nein“.

3.1.3 Modifikatoren

Mengenvereinigung: Die Operation $R := M_1 \cup M_2$ wird durch den Befehl

```
\unionsets{M_1}{M_2}\to{R}
```

realisiert. Ein paar Beispiele sind in Tabelle 2 aufgeführt. Das Ergebnis der Operation ist eine sortierte Menge ohne Duplikate, die die Elemente der Mengen M_1 und M_2 enthält.

Mengendifferenz: Die Operation $R := M_1 - M_2$ (auch $R := M_1 \backslash M_2$ geschrieben) lässt sich mit dem Befehl

```
\minussets{M_1}\minus{M_2}\to{R}
```

durchführen. Ist M_1 eine sortierte Menge, wird auch R sortiert sein. Enthält M_1 Duplikate, enthält R eventuell ebenfalls diese Duplikate. Tabelle 2 enthält einige Beispiele für die Verwendung dieses Befehls.

Die Operation kann man umgangssprachlich wie folgt formulieren: Prüfe für jedes Element e aus M_1, ob $e \in M_2$ gilt. Wenn nein, füge e zu R hinzu. Und genau so wurde es auch straight forward implementiert!

Mengendurchschnitt: Die Operation $R := M_1 \cap M_2$ wird durch den Befehl

```
\intersectsets{M_1}{M_2}\to{R}
```

ermöglicht. Auch hier gilt: Ist M_1 eine sortierte Menge, wird auch R sortiert sein. Enthält M_1 Duplikate, enthält R eventuell ebenfalls diese Duplikate. Tabelle 2 enthält auch Beispiele für die Verwendung dieser Operation.

Befehl	Ergebnis
`\unionsets{\mA}{\mC}\to{\mR}`	„Alice, Bob, Charly, Dean“
`\unionsets{\mB}{\mD}\to{\mR}`	„Alice, Bob, Dean“
`\unionsets{\mL}{\mC}\to{\mR}`	„Bob, Dean“
`\unionsets{\mL}{\mL}\to{\mR}`	„“
`\minussets{\mA}\minus{\mC}\to{\mR}`	„Alice, Charly“
`\minussets{\mD}\minus{\mC}\to{\mR}`	„“
`\minussets{\mD}\minus{\mB}\to{\mR}`	„Dean“
`\minussets{\mA}\minus{\mL}\to{\mR}`	„Alice, Bob, Charly“
`\intersectsets{\mA}{\mB}\to{\mR}`	„Alice, Bob“
`\intersectsets{\mC}{\mB}\to{\mR}`	„Bob“
`\intersectsets{\mB}{\mD}\to{\mR}`	„“
`\intersectsets{\mA}{\mL}\to{\mR}`	„“

Tabelle 2: Mengenoperationen, Beispiele

Die Operation kann man umgangssprachlich wie folgt formulieren: Prüfe für jedes Element e aus M_1, ob $e \in M_2$ gilt. Wenn ja, füge e zu R hinzu. Wenn man dies mit der Formulierung der Mengendifferenz vergleicht, fällt auf, dass der einzige Unterschied im Wörtchen „ja“ besteht. In der Implementierung drückt sich dies durch ein fehlendes `\else` aus. Eigentlich verblüffend einfach, wenn man bedenkt, dass formal $M_1 \cap M_2 \equiv M_1 \backslash (M_1 \backslash M_2)$ gilt, was eine deutlich höhere Komplexität erwarten lässt.

Sortieren: Eine Menge M kann alphabetisch sortiert werden. Dazu verwendet man den Befehl

`\sortset{`M`}{`R`}`.

R enthält danach die sortierte Menge. Die Sortierung erfolgt nach dem Sortierverfahren Bubblesort, einem Verfahren, das sich auch mit TEX ohne größere Verrenkungen umsetzen lässt.

Bei der Sortierung werden die Elemente so verglichen, wie Sie sie angegeben haben, d. h. eventuell enthaltene Kommandos werden nicht expandiert, sondern nach ihrem Namen verglichen (inklusive des Backslash).

Duplikatentfernung: Diese Operation funktioniert *nur* auf sortierten Mengen! Man benötigt sie aber eigentlich auch kaum, da die Erstellung einer neuen Menge mit `\newset` diese Aufgabe automatisch übernimmt (indem sie dieses Makro verwendet). Ich habe mich aber dazu entschlossen, die Operation trotzdem verfügbar zu machen; vielleicht benötigt sie ja tatsächlich einmal jemand. Auf-

gerufen wird die Duplikateleminierung mit:
`\deleteduplicates{`M`}{`R`}`,
wobei R die Ergebnis-Menge darstellt und M die sortierte Menge, deren Duplikate entfernt werden sollen.

3.2 Aufwandsabschätzung

Tabelle 3 beschreibt die Komplexität der Operationen in O-Notation. Dabei werden folgende Annahmen getroffen:

- Die Länge eines Elements einer Menge sei m.
- Die Anzahl der Elemente einer Menge sei n. Werden für eine Operation zwei Mengen benötigt, gibt n_1 die Kardinalität der ersten und n_2 die Kardinalität der zweiten Menge an.
- Zur Vereinfachung sei der Aufwand für die Mustererkennung bei der Parameterübergabe konstant.

Die angegebenen Komplexitätsklassen können dazu dienen, eine Reihe von Mengenoperationen möglichst günstig anzuordnen. Beispielsweise ist es besser, bei `\intersectsets` und `\minussets` als erste Menge die kleinere Menge zu übergeben.

Operation	**Komplexitätsklasse**
Element-Vergleich	m
`\sizeofset`	n
`\listset`	n
`\iselementofset`	1
`\sortset`	$m \cdot n^2$
`\deleteduplicates`	n
`\newset`	$m \cdot n^2$
`\newsetsimple`	1
`\unionsets`	$m \cdot (n_1 + n_2)^2$
`\intersectsets`	n_1
`\minussets`	n_1

Tabelle 3: Komplexitätsklassen der Mengenoperationen