`graphs.sty` — wie es funktioniert, wenn es funktioniert

– Eine kurze Dokumentation –
(Version 1.53)

Frank Drewes, 19.12.2000

1 Vom Allgemeinen ...

Der vorliegende Abschnitt besteht aus generellen Anmerkungen und Erklärungen zur Funktionsweise des durch die Datei `graphs.sty` (zu der außerdem noch drei weitere Dateien `graphs_ps.tex`, `graphs_config.tex` und `graphs.header` gehören) zur Verfügung gestellten Makropakets. Achtung: Das Paket ist nur in Verbindung mit `dvips` (bzw. kompatiblen Programmen) sinnvoll zu nutzen, da es von TEXs `\special`-Befehl in der von `dvips` erwarteten Weise Gebrauch macht. Die einzelnen Kommandos werden im zweiten Abschnitt detaillierter dokumentiert; Beispiele gibt's im dritten.

1.1 Änderungen gegenüber Version 1.43

Zwei Kommandos zum Zeichnen von Bézierkurven (`\bubble` und `\curve`) wurden hinzugefügt, und `\filledarea` wurde in `\area` umbenannt (jedoch bleibt die Kompatibilität mit älteren Versionen gewahrt). Weiterhin wurde ein zweiter Typ von Pfeilspitzen mit dazugehörigem Statuskommando eingeführt, und die Breite der Pfeilspitzen variierbar gemacht. Für einige Statuskommandos wurden außerdem Varianten zur *relativen* Änderung von Werten eingeführt.

1.2 Änderungen gegenüber Version 1.5

Leicht verbesserte (aber immer noch rudimentäre) Fehlererkennung (siehe Abschnitt 1.14). Des weiteren wurde die Angabe des Pfades für die Datei `graphs.header` (die einzige Stelle, an der bei Installation Änderungen nötig sind) zwecks Übersichtlichkeit in eine eigene Datei `graphs_config.tex` ausgelagert.

1.3 Änderungen gegenüber Version 1.51

Es werden jetzt RGB-Farben unterstützt. Dazu ist als Argument für das jeweilige Statuskommando `(`r`,`g`,`b`)` anzugeben, wobei r, g und b den Rot-, Grün und Blauanteil (angegeben durch einen Wert zwischen 0 und 1) festlegen.

1.4 Änderungen gegenüber Version 1.52

Das Kommando `\fillednodesfalse` wurde hinzugefügt; es unterdrückt das Füllen von Knoten mit der aktuellen Farbe (wird selten gebraucht, ist aber manchmal doch nützlich). Das entsprechende Kommando `\fillednodestrue` hat die gegenteilige Wirkung.

1.5 Die Dateien

Die Hauptdatei ist `graphs.sty`, welche als LaTeX2e-Package geladen werden sollte, wenn man die in ihr definierten Makros benutzen will. Diese Datei lädt selbst die Dateien `graphs_ps.tex` und `graphs_config.tex`. Alle drei Dateien sollten sich daher an einem der Orte befinden, die von TeX während der Übersetzung nach zu ladenden Dateien durchsucht werden (z.B. im Verzeichnis `tex/inputs`, in dem auch die anderen `style`-Dateien üblicherweise zu finden sind, oder im Verzeichnis der aktuell zu übersetzenden Datei). Schließlich wird vom *PostScript*-Treiber `dvips` bei der Ausgabe die Datei `graphs.header` benötigt. Der entsprechende Pfad ist in `graphs_config.tex` anzugeben.

1.6 Die Umgebung `graph`

Durch die Datei `graphs.sty` wird eine LaTeX-Umgebung `graph` zur Verfügung gestellt, die, ähnlich der `picture`-Umgebung von LaTeX, Grafik zu erstellen erlaubt. Sie ist insbesondere zur Beschreibung von Graphen und Diagrammen gedacht und (hoffentlich) geeignet. Analog zur Verwendung der `picture`-Umgebung geschieht dies durch

`\begin{graph}`(x_1, x_2)(d_1, d_2)

$\vdots$ (Kommandos)

`\end{graph}`

wobei die Parameter ebenfalls die bekannte Bedeutung haben (und das zweite Paar optional ist).

Die `graph`-Umgebung ist eine echte Erweiterung der LaTeX-`picture`-Umgebung. Daher sind zusätzlich zu den im folgenden beschriebenen Kommandos auch all diejenigen erlaubt, die in der `picture`-Umgebung definiert sind. Eine Variante der `graph`-Umgebung ist die `framegraph`-Umgebung. Letztere ist identisch mit ersterer, außer daß der Graph in der angegebenen Größe in eine `\framebox` eingeschlossen wird. (Das ist hilfreich, wenn man sehen will, ob die angegebene Größe zum definierten Graphen paßt. Einen anderen Sinn hat es nicht.)

1.7 Typen von Zeichenkommandos

Die neuen (d.h. nicht aus `picture` stammenden) Zeichenkommandos lassen sich in vier Gruppen einteilen:

1. Kommandos zum Erzeugen von Knoten,

2. Kommandos zum Erzeugen von Kanten,

3. Kommandos zur Positionierung von Text an Knoten oder Kanten, oder an belieber, durch absolute Koordinaten anzugebender Position. „Text“ meint dabei nicht ausschließlich echten Text, sondern sehr allgemeine Beschriftung: Es ist alles erlaubt, was auch die `picture`-Umgebung als Argument zu `\put` erlaubt.

4. Kommandos zur Erzeugung gefüllter oder ungefüllter, evtl. von Linien umschlossener Bereiche.

Die Kommandos aus den obigen vier Rubriken werden im folgenden *Zeichenkommandos* (im Unterschied zu den weiter unten ebenfalls beschriebenen *Statuskommandos*) genannt.

1.8 Erst Bereiche, dann Kanten, dann Knoten, dann Texte

Zeichenkommandos erzeugen das entsprechende Bild nicht unmittelbar. Anstatt dessen werden intern die nötigen Informationen in separaten Listen (für gefüllte Bereiche, Knoten, Kanten und Texte) gesammelt. Erst am Ende der Umgebung werden zuerst die Linienzüge, dann Kanten, dann Knoten und dann Texte ausgegeben. Das Resultat ist, daß, unabhängig von der Reihenfolge der Kommandos, Texte immer über Knoten, Kanten und Bereichen liegen, Knoten immer über Kanten und Bereichen, und Kanten immer über Bereichen. Jedoch ist für Knoten (Kanten, Texte, Bereiche) untereinander die Reihenfolge relevant: Ein später an überlappender Position definierter Knoten überdeckt den vorher definierten. Analoges gilt für Kanten, Beschriftungen und Bereiche.

Da alle Definitionen erst am Schluß wirksam werden, gibt es im Grunde noch eine Ebene „unter“ den Bereichen: Alle `picture`-Kommandos werden unmittelbar ausgeführt, kommen also vor allen anderen zur Geltung.

1.9 Linienbreite, Füllfarbe, und Konsorten

Für Linienbreite, Füllfarbe[1] und einiges mehr existieren Voreinstellungen, die global und lokal mit Hilfe einiger Makros, genannt *Statuskommandos*, geändert werden können.

Um lokale Einstellungen vornehmen zu können, kann *jedes* Zeichenkommando mit einem optionalen Parameter versehen werden, in dem die entsprechenden Statuskommandos aufgelistet werden. Die jeweiligen Einstellungen gelten lokal für den Aufruf desjenigen Zeichenkommandos, in dessen optionalem Parameter sie erfolgen. (Natürlich wirkt sich nicht jede Änderung auf jedes Zeichenkommando aus — für einen Knoten ist eine lokale Änderung der Größe von Pfeilspitzen bedeutungslos, aber auch unschädlich.)

Eine globale Änderung geschieht durch Verwendung eines Statuskommandos in der `graph`-Umgebung selbst (außerhalb des Parameters eines Zeichenkommandos), dann gilt sie für diese eine Umgebung, oder außerhalb der Umgebung, dann gilt sie für alle folgenden Graphen. Aufgrund der Tatsache, daß das eigentliche Zeichnen immer am Schluß geschieht, wirken sich globale Einstellungen innerhalb einer Umgebung immer auf die ganze Umgebung aus, d.h. auch auf Zeichenkommandos, die vorher aufgeführt werden. Dies macht im allgemeinen die mehrmalige globale Änderung einer Einstellung innerhalb eines Graphen sinnlos. Analoges gilt für die Änderung von `\unitlength` (was man, um verwirrende Effekte zu vermeiden, auf globaler Ebene *innerhalb* eines Graphen sowieso nicht ändern sollte).

1.10 Einheiten

Soweit die Parameter von Kommandos Größen beschreiben ist deren Einheit `\unitlength` (Einheiten werden also *nicht* mit angegeben), mit Ausnahme des Statuskommandos für Linienstrichelung. Letzteres hat „Punkt“ als Einheit.

1.11 Benutzung von `german.sty`

Beim Schreiben von deutschen Texten mit `german.sty` muß vor einer `graph`-Umgebung `\originalTeX` aufgerufen werden, da sich die im `german.sty` definierte Spezialbehand-

[1] „Füllfarbe“ ist natürlich keine echte Farbe, sondern ein Grauwert.

lung von `"` nicht mit dem `special`-Kommando von TeX verträgt (wie mir beim Schreiben dieser Anleitung drastisch klargemacht wurde ...). Hinter dem `\end{graph}` kann mit `\germanTeX` wieder zu Deutsch umgeschaltet werden.

1.12 Ansehen und drucken

Intern werden die Grafikbefehle in die von `dvips` verstandenen `\special{`-Kommandos umgesetzt (mit Ausnahme der Befehle für Beschriftungen). Zum Ausdrucken sollte daher nur `dvips` verwendet werden. Außerdem bewirkt dies, daß dvi-previewer (gibt's dafür ein passendes deutsches Wort?) nur den Text zeigen. Zum Ansehen ist es daher nötig, sich die *PostScript*–Ausgabe von `dvips` — umgeleitet in eine Datei — mittels eines *PostScript*–previewers anzeigen zu lassen.

1.13 Unterdrückung des Zeichnens von Graphen

Manchmal ist es angenehm, das Zeichnen von Graphen unterdrücken zu können, weil dies den LaTeX-Lauf beschleunigen kann, wenn viele und komplizierte Graphen in einem Text enthalten sind. Dazu gibt es die Kommandos `\drawgraphsfalse` und `\drawgraphstrue` (mit der offensichtlichen Bedeutung); letzteres ist Voreinstellung. Bei `\drawgraphsfalse` erscheint statt des Graphen ein Rechteck der entsprechenden Größe mit der Inschrift *graph*. Außerdem kann der "package" `graphs` die Option `draft` in der üblichen Weise übergeben werden. Dies hat die gleiche Wirkung wie ein Aufruf `\drawgraphsfalse` am Anfang des Dokuments.

1.14 Fehlerbehandlung

Die Fehlerbehandlung ist (noch?) recht rudimentär. Sie beschränkt sich mehr oder weniger auf zwei Dinge: Erstens, zu prüfen, ob benutzte Knoten auch definiert werden und zweitens, ob die Knoten, zwischen denen Kanten gezogen werden, unterschiedliche Koordinaten haben. Bei anderen Fehlern hilft nur scharfes Hinsehen und Vergleichen mit dieser Dokumentation. Am häufigsten unterlaufen einem wohl falsche Parameterangaben, was meist recht fatale Folgen hat. Das oben angesprochene Problem mit `german.sty` führt auch oft zu Verwirrung. Weiterhin kann es sein, daß erst der *PostScript*-Interpreter einen Fehler liefert, was zum Beispiel durch einen Aufruf der Form `\bow{K1}{K2}{xyz}` provoziert werden kann. Die Zeichenkette „xyz“ wird nämlich unbesehen ins *PostScript* übernommen, aber dort wird an dieser Stelle eine Zahl erwartet (siehe unten).

2 ... zum Besonderen

Nun zu den einzelnen Kommandos. Es folgen zwei Tabellen mit den Status- und Zeichenkommandos, jeweils gefolgt von ausführlicheren Beschreibungen.

2.1 Statuskommandos

`\graphnodesize{`x`}`	Knotendurchmesser x Einheiten. Voreinstellung: 0.2

`\graphnodecolour{`x`}`
`\graphnodecolour(`r`,`g`,`b`)`
`\graphnodecolour+{`f`}`
	Füllfarbe Schwarz $= 0 \leq x \leq 1 =$ Weiß, bzw. drei separate Werte für RGB-Farben. Voreinstellung: Schwarz

`\fillednodestrue`, `\fillednodesfalse`	Knoten werden gefüllt, oder nicht. Voreinstellung: true
`\graphlinewidth{`x`}` `\graphlinewidth*{`f`}`	Linienbreite x Einheiten. Voreinstellung: 0.02

`\graphlinecolour{`x`}`
`\graphlinecolour(`r`,`g`,`b`)`
`\graphlinecolour+{`f`}`
	Linienfarbe Schwarz $= 0 \leq x \leq 1 =$ Weiß, bzw. drei separate Werte für RGB-Farben. Voreinstellung: Schwarz

`\graphlinedash{`x_1 x_2 $\ldots$`}`	Linienstrichelung. Voreinstellung: Durchgehend
`\grapharrowlength{`x`}` `\grapharrowlength*{`f`}`	Pfeilspitzenlänge x Einheiten. Voreinstellung: 0.3
`\grapharrowwidth{`x`}` `\grapharrowwidth*{`f`}`	Pfeilspitzenbreite relativ zur -länge. Voreinstellung: 0.5
`\grapharrowtype{`x`}`	Typ (1 oder 2) der Pfeilspitzen. Voreinstellung: 1
`\autodistance{`x`}`	Entfernungsfaktor für automatisch plazierten Text. Voreinstellung: 1.3
`\enlargeboxes{`x`}`	Text horizontal und vertikal x Einheiten mehr Platz geben. Voreinstellung: 0.1
`\opaquetexttrue`, `\opaquetextfalse`	Text deckt Unterliegendes ab, oder nicht. Voreinstellung: true
`\filledareastrue`, `\filledareasfalse`	Bereiche werden gefüllt, oder nicht. Voreinstellung: true

`\graphfillcolour{`x`}`
`\graphfillcolour(`r`,`g`,`b`)`
`\graphfillcolour+{`f`}`
	Füllfarbe Schwarz $= 0 \leq x \leq 1 =$ Weiß, bzw. drei separate Werte für RGB-Farben. Voreinstellung: 0.5

Die $*$- bzw. $+$-Varianten der Statuskommandos verändern den aktuellen Wert *relativ*. Hierbei ist f ein (positiver) Faktor. Die $+$-Varianten hellen den Grauwert bei Argumenten > 1 auf; bei Argumenten zwischen 0 und 1 dunkeln sie ab. Zum Beispiel ergibt das Argument 3 dreifache Helligkeit (beim aktuellen Wert von 0.7 ergibt sich 0.9), und das Argument 0.2 reduziert die Helligkeit auf ein Fünftel (beim aktuellen Wert von 0.7 ergibt sich 0.14).

2.2 Die Statuskommandos im einzelnen:

- `\graphnodesize{`x`}`

  Hier ist x der Knotendurchmesser in Einheiten von `\unitlength`. Der Knotendurchmesser ist unabhängig von der Linienbreite (solange diese ihn nicht überschreitet), d.h. das Innere wird kleiner, wenn die Linienbreite größer wird.

- `\graphnodecolour{`x`}` bzw. `\graphnodecolour(`r`,`g`,`b`)`

  Die Parameter x, r, b, g müssen zwischen 0 und 1 liegen. Mit dem entsprechenden Grau- bzw. Farbwert werden Knoten ausgefüllt.

- `\fillednodestrue`, `\fillednodesfalse`

  Die Kommandos zum Zeichnen von Knoten produzieren mit dem festgelegten Grauwert gefüllte Knoten falls `\fillednodestrue`, und sonst ungefüllte. Letzteres ist normalerweise unerwünscht. Da Kanten immer vom Mittelpunkt des Knotens ausgehend gezeichnet werden, ist die Füllfarbe Weiß üblicherweise vorzuziehen. Allerdings gibt es Sonderfälle, in denen `\fillednodesfalse` für spezielle Effekte sinnvoll Verwendung finden kann.

- `\graphlinewidth{`x`}`

  Hier bezeichnet x die Dicke von Linien, d.h. von Kanten und von Knotenumrissen, in Einheiten von `\unitlength`. Wie durch *PostScript* definiert, bewirkt 0 nicht wirklich Linienbreite 0, sondern die geringstmögliche darstellbare Breite (geräteabhängig!).

- `\graphlinecolour{`x`}` bzw. `\graphlinecolour(`r`,`g`,`b`)`

  Der Grau- bzw. Farbwert für Linien (d.h. von Kanten und von Knotenumrissen), analog zu `\graphnodecolour`.

- `\graphlinedash{`x_1 x_2 $\cdots$ x_n`}`

  Die Strichelung von Linien (d.h. von Kanten und von Knotenumrissen). Die x_i sind positive Zahlen; die Liste wird zyklisch wie folgt interpretiert: Linie von $x_1$pt Länge, Lücke von $x_2$pt Länge, Linie von $x_3$pt Länge, usw. Eine leere Liste bedeutet „durchgehende Linie“; sonst muß — aus naheliegenden Gründen — mindestens ein Listeneintrag ungleich 0 sein.

  Beispiele:

  - 1 bewirkt 1pt Linie, 1pt Lücke, 1pt Linie, 1pt Lücke, usw.
  - 1 2 bewirkt 1pt Linie, 2pt Lücke, 1pt Linie, 2pt Lücke, usw.
  - 3 2 1 bewirkt 3pt Linie, 2pt Lücke, 1pt Linie, 3pt Lücke, usw.

- `\grapharrowlength{`x`}`

  Die Länge von Pfeilspitzen beträgt x Einheiten.

- `\opaquetexttrue`, `\opaquetextfalse`

  Zeichenkommandos, die Text, d.h. Beschriftungen produzieren, unterlegen diesen mit einem weißen Rechteck entsprechender Größe, falls `\opaquetexttrue` gilt. Dies macht z.B. auch die Beschriftung des Inneren schwarz ausgefüllter Knoten möglich.

- `\autodistance{`x`}`

  Der Faktor, um den automatisch an Knoten plazierte Textboxen zur Seite und nach oben/unten verschoben werden (gesehen vom Mittelpunkt aus und relativ zur Größe des Knotens). Bei einem Faktor von 1 liegt die dem Knoten nahe Ecke auf dem Rand.

- `\grapharrowwidth{`x`}`

  Die Breite der Basis von Pfeilspitzen beträgt das x-fache der Länge.

- `\grapharrowtype{`x`}`

  Der Typ der verwendeten Pfeilspitzen. Typ 1 ist ein Dreieck, während bei Typ 2 die Seiten etwas nach innen gebogen werden.

- `\opaquetexttrue`, `\opaquetextfalse`

  Zeichenkommandos, die Text, d.h. Beschriftungen produzieren, unterlegen diesen mit einem weißen Rechteck entsprechender Größe, falls `\opaquetexttrue` gilt. Dies macht z.B. auch die Beschriftung des Inneren schwarz ausgefüllter Knoten möglich.

- `\autodistance{`x`}`

  Der Faktor, um den automatisch an Knoten plazierte Textboxen zur Seite und nach oben/unten verschoben werden (gesehen vom Mittelpunkt aus und relativ zur Größe des Knotens). Bei einem Faktor von 1 liegt die dem Knoten nahe Ecke auf dem Rand.

- `\enlargeboxes{`x`}`

  Die Größe der Rechtecke, die Beschriftungen unterlegt werden (siehe oben) ergibt sich aus der Ausdehnung des Textes plus x Einheiten in beiden Dimensionen. Dies verhindert, daß der Text zu sehr „eingequetscht“ erscheint.

- `\filledareastrue`, `\filledareasfalse`

  Die Kommandos zum Zeichnen von Linienzügen produzieren mit dem festgelegten Grauwert gefüllte Linienzüge falls `\filledareastrue`, und sonst ungefüllte.

- `\graphfillcolour{`x`}` bzw. `\graphfillcolour(`r`,`g`,`b`)`

  Der Grau- bzw. Farbwert für die Füllung von Bereichen, die mit `\area` produziert werden. Wirkungslos, falls `\filledareasfalse`.

2.3 Zeichenkommandos

`\squarenode{Name}`(x,y)	Quadratischer Knoten "Name" an Position (x, y).
`\roundnode{Name}`(x,y)	Runder Knoten "Name" an Position (x, y).
`\rectnode{Name}`$[b,h](x,y)$	Rechteckiger Knoten "Name" der Breite b und Höhe h an Position (x, y).
`\textnode{Name}`(x,y)`{Text}`	Rechteckiger Knoten "Name" an Position (x, y) mit Beschriftung (Größe entsprechend Beschriftung).
`\edge{`Name_1`}{`Name_2`}`	Ungerichtete Kante von Name_1 zu Name_2.
`\diredge{`Name_1`}{`Name_2`}`	Gerichtete Kante von Name_1 zu Name_2.
`\bow{`Name_1`}{`Name_2`}` `{Auslenkung}`	Gebogene ungerichtete Kante von Name_1 zu Name_2 mit relativem Auslenkungsfaktor.
`\dirbow{`Name_1`}{`Name_2`}` `{Auslenkung}`	Gebogene gerichtete Kante von Name_1 zu Name_2 mit relativem Auslenkungsfaktor.
`\loopedge{Name}`(x_1,y_1) (x_2,y_2) oder `\loopedge{Name}`$\{\alpha\}(x,y)$	Ungerichtete Schleife an Name, wobei entweder (x_1, y_1), (x_2, y_2) relative Koordinaten der Endpunkte der aus- und eingehenden Teilkanten sind, oder α ihren Öffnungswinkel darstellt und (x, y) Mittelachse und Länge festlegt.
`\dirloopedge{Name}`(x_1,y_1) (x_2,y_2) oder `\dirloopedge{Name}`$\{\alpha\}(x,y)$	Gerichtete Schleife an Name, wobei entweder (x_1, y_1), (x_2, y_2) relative Koordinaten der Endpunkte der aus- und eingehenden Teilkanten sind, oder α ihren Öffnungswinkel darstellt und (x, y) Mittelachse und Länge festlegt.
`\autonodetext{Name}` `[n|e|s|w|ne|nw|se|sw]{Text}`	Autom. Positionierung von Text an Name mit optionalem Parameter „Himmelsrichtung".
`\nodetext{Name}`(x,y) `{Text}`	Positionierung von Text an Name mit relativen Koordinaten (x, y) (optional).
`\edgetext{`Name_1`}{`Name_2`}` `{Text}`	Positionierung von Text in der Mitte zwischen Name_1 und Name_2.
`\bowtext{`Name_1`}{`Name_2`}` `{Auslenkung}{Text}`	Positionierung von Text in der Mitte des durch die Parameter beschriebenen Bogens.
`\freetext`(x,y)`{Text}`	Positionierung von Text mit absoluten Koordinaten (x, y).

Zusätzlich zu den oben aufgelisteten Parametern kann jedes der Kommandos noch mit einem $(n+1)$-ten optionalen Parameter `[···]` versehen werden, in dem mit Hilfe der Statuskommandos lokale, nur für dieses Kommando gültige Einstellungen vorgenommen werden.

`\area(`x_1`,`y_1`)` `{(`x_2`,`y_2`[,`d_2`])(`x_3`,`y_3`[,`d_3`])`$\cdots$`}`	(Eventuell gefüllter) Bereich mit absoluten Koordinaten der Eckpunkte. Startkoordinaten (x_1, y_1), von (x_i, y_i) zu (x_{i+1}, y_{i+1}) mit Auslenkungsfaktor d_i.
`\bubble{`f`}{` `(`x_1`,`y_1`)(`x_2`,`y_2`)`$\cdots$`}`	(Eventuell gefüllte) Bézierkurve durch (x_1, y_1), (x_2, y_2), $\ldots$, wobei f den Abstand der Kontrollpunkte von den eigentlichen Punkten in Vielfachen der Segmentlänge angibt.
`\curve{`f`}{` `(`x_1`,`y_1`,`α_1`)[`$f_{1,1}$`,`$f_{1,2}$`]` `(`x_2`,`y_2`,`α_2`)[`$f_{2,1}$`,`$f_{2,2}$`]`$\cdots$`}`	(Eventuell gefüllte) Bézierkurve durch (x_1, y_1), (x_2, y_2), $\ldots$ mit Kontrollpunkten definiert durch f (normaler Abstand in Vielfachen der Segmentlänge), α_i (Winkel), $f_{i,j}$ (opt. für anderen Abstand).
Zusätzlich zu den oben aufgelisteten Parametern kann jedes der Kommandos noch mit einem $(n+1)$-ten optionalen Parameter `[`$\cdots$`]` versehen werden, in dem mit Hilfe der Statuskommandos lokale, nur für dieses Kommando gültige Einstellungen vorgenommen werden.	

2.4 Die Zeichenkommandos im einzelnen:

- `\squarenode{`Name`}(`x`,`y`)`

  Ein quadratischer Knoten Name mit Mittelpunkt an Position (x, y). Name kann alle Zeichen enthalten, die TEX zwischen `\csname` und `\endcsname` erlaubt. Insbesondere schließt dies Buchstaben und Ziffern ein.

  Die (durch `\graphnodesize` veränderbare) Knotengröße ist unabhängig von der Linienbreite (die bei Knoten die Breite der Umrißlinie bezeichnet). Eine größere Linienbreite führt zu einem kleineren Inneren der Knoten.

  Abhängigkeiten:
  `\graphnodesize`, `\graphnodecolour`, `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\fillednodestrue/false`

- `\roundnode{`Name`}(`x`,`y`)`

  Wie `\squarenode`, nur daß stattdessen ein runder Knoten produziert wird.

  Abhängigkeiten:
  `\graphnodesize`, `\graphnodecolour`, `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\fillednodestrue/false`

- `\rectnode{`Name`}[`b`,`h`](`x`,`y`)`

  Wie `\squarenode`, nur daß stattdessen ein rechteckiger Knoten produziert wird, dessen Breite und Höhe explizit durch b und h gegeben sind (wie immer in Vielfachen von `\unitlength`).

  Abhängigkeiten:
  `\graphnodecolour`, `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\fillednodestrue/fa`

- `\textnode{Name}(`x`,`y`){Text}`

  Wie `\rectnode`, wobei aber die Größe hier definierten Knotens durch den Text (plus `\enlargeboxes` plus `\graphlinewidth`) gegeben ist. Intern wird der Text mittels `\autonodetext` produziert (siehe unten).

  Achtung: Dieses Kommando hat eine private Voreinstellung von `\graphnodecolour`, nämlich 1 (= Weiß). Dies erscheint wegen der ebenfalls privaten Einstellung von `\opaquetextfalse` bei `\autonodetext` sinnvoll und kann im optionalen Parameter gegebenenfalls geändert werden.

  Abhängigkeiten:
  `\graphnodecolour` (im optionalen Parameter),
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\opaquetexttrue/false` (im optionalen Parameter), `\enlargeboxes`, `\fillednodestrue/false`

- `\edge{`Name$_1$`}{`Name$_2$`}`

  Eine direkte, ungerichtete Kante von Knoten Name$_1$ zu Knoten Name$_2$.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`

- `\diredge{`Name$_1$`}{`Name$_2$`}`

  Wie `\edge`, nur gerichtet.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\grapharrowlength`, `\grapharrowwidth`, `\grapharrowtype`

- `\bow{`Name$_1$`}{`Name$_2$`}{Auslenkung}`

  Eine ungerichtete, gebogene Kante von Knoten Name$_1$ zu Knoten Name$_2$. Dabei bestimmt der Auslenkungsfaktor das Ausmaß der Verschiebung des Mittelpunktes der Kante nach links, relativ zur Entfernung der beiden Knoten voneinander. Ein negativer Faktor bewirkt eine entsprechende Verschiebung nach rechts. (Relative Auslenkungsfaktoren haben gegenüber absoluten Werten den Vorteil, daß die Form einer Kante unabhängig von den Positionen der Knoten ist. Zum Beispiel führt der Faktor 0.5 immer zu einer Kante, die einen Halbkreis bildet. So können die Auswirkungen nachträglicher Knotenverschiebungen besser abgeschätzt werden.)

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`

- `\dirbow{`Name$_1$`}{`Name$_2$`}{Auslenkung}`

  Wie `\bow`, nur gerichtet.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\grapharrowlength`, `\grapharrowwidth`, `\grapharrowtype`

- `\loopedge{Name}(`x_1`,`y_1`)(`x_2`,`y_2`)`
  oder
  `\loopedge{Name}{`α`}(`x`,`y`)`

Schleife an Knoten Name. In der ersten Form werden die Koordinaten (x_1, y_1) und (x_2, y_2) als relative Koordinaten der Endpunkte zweier von Name ausgehender Halbkanten interpretiert. Diese werden beim Zeichnen mit einer Kurve verbunden, um eine Schleife zu erhalten.

Die zweite Form legt durch α den Winkel fest, den beide Halbkanten einschließen und mit (x, y) (ein Punkt relativ zur Knotenposition) die Mittelachse und Länge der beiden. Die zweite Form ist also bequemer, weil sie auf einfache Weise symmetrische Schleifen zu erzeugen erlaubt, aber auch eingeschränkter, weil hierauf beschränkt.

Bei beiden Formen wird immer von dem kleineren Winkel (also der ≤ 180 Grad) ausgegangen; eine Angabe von 200 Grad bei der zweiten Form ist also äquivalent zu 160 Grad plus einer Punktspiegelung von (x, y) am Knotenmittelpunkt.

Abhängigkeiten:
`\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`

- `\dirloopedge{`Name`}(`x_1`,`y_1`)(`x_2`,`y_2`)`

  oder

  `\dirloopedge{`Name`}{`α`}(`x`,`y`)`

  Wie `\loopedge`, nur gerichtet. Die Pfeilspitze erscheint an der durch (x_2, y_2) angegebenen Halbkante, beziehungsweise bei der zweiten Form entgegen dem Uhrzeigersinn.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinecolour`, `\graphlinedash`, `\grapharrowlength`, `\grapharrowwidth`, `\grapharrowtype`

- `\autonodetext{`Name`}[n|e|s|w|ne|nw|se|sw]{`Text`}`

  Automatische Positionierung von Text am Knoten Name. Falls vorhanden, wird der optionale Parameter als Himmelsrichtung interpretiert und der Text erscheint an der entsprechenden Seite des Knotens. Sonst erscheint er innerhalb (d.h. im Mittelpunkt) des Knotens.

  „Text“ meint, wie bei den folgenden Kommandos auch, all das, was dem `\put`-Kommando der LaTeX-`picture`-Umgebung als Argument übergeben werden kann. Es ist also nahezu alles erlaubt, nötigenfalls in `\parbox` oder ähnlichem eingeschlossen. Insbesondere kann ein Graph als Beschriftung verwendet werden. Dies hat den Vorteil, daß ganze Teilgraphen in ihrer Position relativ zu bestimmten Knoten oder Kanten definiert werden können. Werden diese hinterher verschoben, verschieben sich die Teilgraphen automatisch mit, ohne daß etwas getan werden muß. Als Beispiel betrachte man das Kommando

```
\autonodetext{K}[se]{%
  \begin{graph}(1,1)
    \graphnodesize{.1}
    \squarenode{sqa}(0,1)
    \squarenode{sqb}(1,1)
    \edge{sqa}{sqb}
  \end{graph}},
```

das als Text rechts unten vom Knoten `K` einen Graphen mit Knotendurchmesser 1mm, zwei Knoten und eine Kante dazwischen enthaltend, angibt. Wird der Knoten `K` im umgebenden Graphen verschoben, so behält der kleinere Graph seine Position relativ zu diesem Knoten bei. Analoges gilt auch für die anderen Textkommandos (siehe unten).

Im Unterschied zu allen anderen Textkommandos hat `\autonodetext` eine feste Voreinstellung von `\opaquetext`, und zwar `\opaquetextfalse`. Diese wird durch die globale Einstellung nicht beeinflußt. Der Grund für diese Festlegung ist, daß bei `\autonodetext` die Einstellung `\opaquetexttrue` — ansonsten die Standardeinstellung — meist hinderlich ist, denn sie führt in vielen Fällen zu „angeknabberten" Knoten. *Lokal* kann diese Einstellung natürlich wie gewohnt überschrieben werden (also durch einen vierten Parameter `[\opaquetexttrue...]`).

Abhängigkeiten:
`\opaquetexttrue/false` (im optionalen Parameter), `\autodistance`,
`\enlargeboxes` (falls `\opaquetexttrue`)

- `\nodetext{`Name`}(`x`,`y`){`Text`}`

  Positionierung von Text an Koordinaten (x, y) relativ zum Knoten Name. Die Koordinaten sind optional; fehlen sie, wird $(0, 0)$, also der Mittelpunkt des Knotens, eingesetzt. Ansonsten gilt das zu `\autonodetext` Gesagte. (Beachte allerdings den Unterschied zwischen `\autonodetext{`Name`}{`Text`}` und `\nodetext{`Name`}{`Text`}` bezüglich der Einstellung von `\opaquetext`.)

  Abhängigkeiten:
  `\opaquetexttrue/false`, `\enlargeboxes` (falls `\opaquetexttrue`)

- `\edgetext{`Name$_1$`}{`Name$_2$`}{`Text`}`

  Positionierung von Text in der Mitte zwischen Knoten Name$_1$ und Name$_2$. Es ist hierzu nicht nötig, daß zwischen den Knoten wirklich eine Kante existiert.

  Ansonsten gilt das zu `\autonodetext` Gesagte analog.

  Abhängigkeiten:
  `\opaquetexttrue/false`, `\enlargeboxes` (falls `\opaquetexttrue`)

- `\bowtext{`Name$_1$`}{`Name$_2$`}{`Auslenkung`}{`Text`}`

  Positionierung von Text in der Mitte der durch die ersten drei Parameter beschriebenen gebogenen Kante. Es ist hierzu nicht nötig, daß zwischen den Knoten wirklich eine entsprechende Kante existiert.

  Ansonsten gilt das zu `\autonodetext` Gesagte analog.

  Abhängigkeiten:
  `\opaquetexttrue/false`, `\enlargeboxes` (falls `\opaquetexttrue`)

- `\freetext(`x`,`y`){`Text`}`

  Positionierung von Text an absoluten Koordinaten (x, y).

  Ansonsten gilt das zu `\autonodetext` Gesagte analog.

  Abhängigkeiten:
  `\opaquetexttrue/false`, `\enlargeboxes` (falls `\opaquetexttrue`)

- `\area(`x_1`,`y_1`){(`x_2`,`y_2[`,`d_2]`)(`x_3`,`y_3[`,`d_3]`)`$\cdots$`}`

  Definition eines (bei `\filledareastrue` mit dem aktuellen, durch `\graphfillcolour` veränderbaren Grauwert gefüllten) Linienzugs. Alle Koordinaten sind absolut. Startkoordinaten sind (x_1, y_1), von da geht es zu (x_2, y_2), dann zu (x_3, y_3), und so fort. Dabei bestimmt d_i den Auslenkungsfaktor des jeweiligen Liniensegments, analog zum dritten Argument von `\bow`.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinedash`, `\graphlinecolour`, `\filledareastrue/false`, `\graphfillcolour` (falls `\filledareastrue`)

- `\bubble{`f`}{(`x_1`,`y_1`)(`x_2`,`y_2`)`$\cdots$`}`

  Wie `\area`, wobei aber der Linienzug eine Bézierkurve durch die gegebenen Punkte ist. Das erste Argument ist ein Faktor, der bestimmt, wie weit die Kontrollpunkte der Bézierkurve von den eigentlichen Punkten entfernt sind. Als Entfernung wird dabei der Abstand eines Punktes zum nächsten, multipliziert mit f, genommen.

  Mittels `\bubble` erzeugte Linienzüge werden immer automatisch geschlossen.

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinedash`, `\graphlinecolour`, `\filledareastrue/false`, `\graphfillcolour` (falls `\filledareastrue`)

- `\curve{`f`}{(`x_1`,`y_1`,`α_1`)[`$f_{1,1}$`,`$f_{1,2}$`](`x_2`,`y_2`,`α_2`)[`$f_{2,1}$`,`$f_{2,2}$`]`$\cdots$`}`

  Eine flexiblere, aber komplexere Möglichkeit, Linienzüge mittels Bézierkurven zu erzeugen. Eine Bézierkurve passiert eine Reihe von Punkten, wobei jedem Segment zwei Kontrollpunkte zugeordnet sind. Die Kurve verläßt den Anfangspunkt des Segmentes in Richtung des ersten Kontrollpunktes und trifft auf den Endpunkt aus Richtung des zweiten (d.h. die Tangenten der Kurve in den jeweiligen Punkten gehen durch die Kontrollpunkte). Bildlich gesprochen ziehen die Kontrollpunkte die Kurve an, so daß weiter entfernte Kontrollpunkte die Kurve auch weiter von der direkten Strecke zum nächsten Punkt ablenken.

  Die Koordinaten (x_i, y_i) (im folgenden P_i) sind absolut; sie geben die eigentlichen Punkte der Kurve an. Die beiden dem Segment $\overline{P_i P_{i+1}}$ zugeordneten Kontrollpunkte sind durch durch P_i, α_i und $f_{i,1}$ bzw. durch P_{i+1}, α_{i+1} und $f_{i,2}$ bestimmt. Dabei wird $f_{i,j} = f$ gesetzt, wenn für dieses Segment keine $f_{i,j}$ angegeben sind. Der erste Kontrollpunkt (für den zweiten geht's analog) ist wie folgt festgelegt: Der Punkt liegt auf derjenigen Geraden durch P_i, die mit der x-Achse den Winkel α_i einschließt. Die Entfernung des Punktes zu P_i ist die Länge des Segmentes $\overline{P_i P_{i+1}}$, multipliziert mit f_i. Dabei liegt dieser erste Kontrollpunkt in dem durch α_i bestimmten Quadranten, während der zweite jeweils auf der anderen Seite liegt.

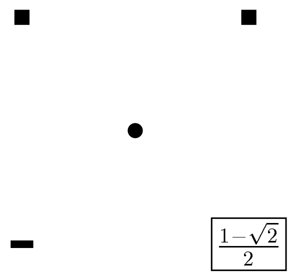
  Ein durch `\curve` erzeugter Linienzug wird *nicht* automatisch geschlossen. (Daraus folgt, daß ein optionales Argument nach dem letzten Punkt folgenlos ist, da es keinem Segment zugeordnet wird.)

  Abhängigkeiten:
  `\graphlinewidth`, `\graphlinedash`, `\graphlinecolour`, `\filledareastrue/false`, `\graphfillcolour` (falls `\filledareastrue`)

3 Beispiele

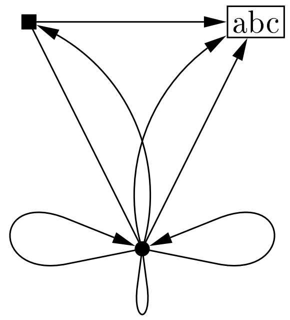
3.1 Knoten, nichts als Knoten

```
\begin{graph}(4,4)(-2,-2)
  \squarenode{Sq1}(-1.5,1.5)
  \squarenode{Sq2}(1.5,1.5)
  \textnode{Te}(1.5,-1.5){$\frac{1-\sqrt 2}2$}
  \rectnode{Re}[.3,.1](-1.5,-1.5)
  \roundnode{Ro}(0,0)
\end{graph}
```



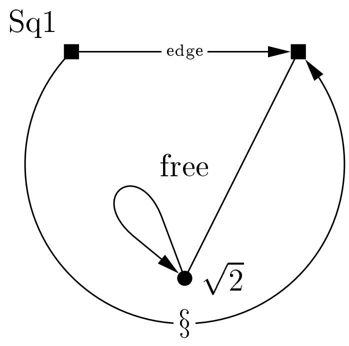
3.2 . . . und ein paar Kanten

```
\begin{graph}(4,4)(-2,0)
  \squarenode{Sq}(-1.5,3.5)
  \textnode{Te}(1.5,3.5){abc}
  \roundnode{Ro}(0,.5)
  \diredge{Sq}{Te}
  \edge{Sq}{Ro} \diredge{Ro}{Te}
  \dirbow{Ro}{Sq}{-.2}
  \dirbow{Ro}{Te}{.2}
  \dirloopedge{Ro}(-.8,-.2)(-1.2,.4)
  \dirloopedge{Ro}(.8,-.2)(1.2,.4)
  \loopedge{Ro}{15}(0,-.5)
\end{graph}
```



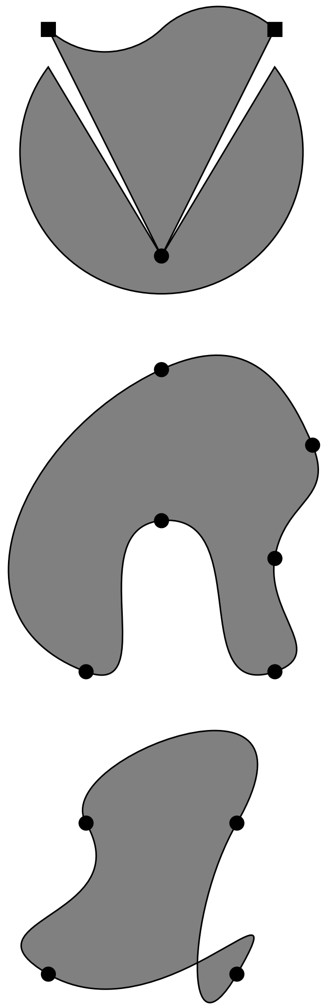
3.3 Nun ein wenig mehr Text:

```
\begin{graph}(4,4)(-2,0)
  \squarenode{Sq1}(-1.5,3.5)
  \squarenode{Sq2}(1.5,3.5)
  \roundnode{Ro}(0,.5)
  \diredge{Sq1}{Sq2}
  \dirloopedge{Ro}{30}(-.5,.7)
  \edge{Sq2}{Ro}
  \dirbow{Sq1}{Sq2}{-1.2}
  \autonodetext{Sq1}[nw]{Sq1}
  \nodetext{Ro}(.5,0){$\sqrt{2}$}
  \edgetext{Sq1}{Sq2}{\tiny edge}
  \bowtext{Sq1}{Sq2}{-1.2}{\S}
  \freetext(0,2){free}
\end{graph}
```



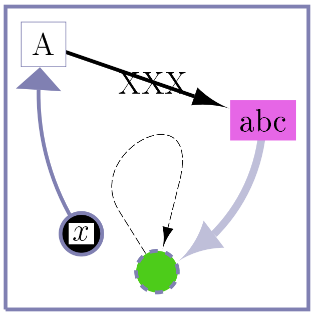
3.4 Linienzüge

```
\begin{graph}(4,13.5)(-2,-9.5)
  \squarenode{Sq1}(-1.5,3.5)
  \squarenode{Sq2}(1.5,3.5)
  \roundnode{Ro}(0,.5)
  \area(0,.5){%
   (-1.5,3.5)(0,3.5,-.2)(1.5,3.5,.2)(0,.5)}
  \area(0,.5){%
   (-1.5,3)(1.5,3,-1)(0,.5)}
  \roundnode a(-1,-5)
  \roundnode b(0,-3)
  \roundnode c(1.5,-5)
  \roundnode d(1.5,-3.5)
  \roundnode e(2,-2)
  \roundnode f(0,-1)
  \bubble{.5}{%
   (-1,-5)(0,-3)(1.5,-5)(1.5,-3.5)(2,-2)(0,-1)}
  \roundnode g(-1.5,-9)
  \roundnode h(1,-9)
  \roundnode i(1,-7)
  \roundnode j(-1,-7)
  \curve{.7}{(-1.5,-9,-30)(1,-9,-30)%
   (1,-7,60)[1.3,.5](-1,-7,-60)(-1.5,-9,-30)}
\end{graph}
```



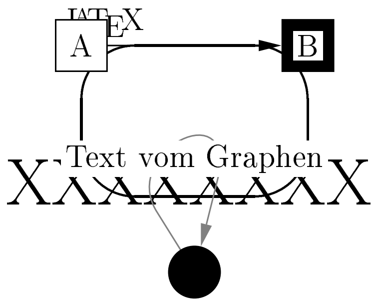
3.5 Fehlen noch die Statuskommandos ...

```
\begin{graph}(4,4)(-2,0)
  \graphnodesize{.6}\graphlinecolour{.5}
  \graphlinewidth*{2.5}\filledareasfalse
  \grapharrowtype{2}
  \squarenode{Sq}(-1.5,3.5)[
    \graphlinewidth{.01}
    \graphnodecolour{1}]
  \textnode{Te}(1.4,2.5){abc}[
    \graphlinecolour(.9,.4,.9)
    \graphnodecolour(.9,.4,.9)
    \enlargeboxes{.2}]
  \roundnode{Ro1}(0,.5)[
    \graphnodecolour(1,.3,0)
    \graphlinedash{3}]
  \roundnode{Ro2}(-1,1)
  \diredge{Sq}{Te}[
    \grapharrowlength{.5}
    \graphlinecolour{0}]
  \dirbow{Ro2}{Sq}{.1}[
    \grapharrowwidth{2}
    \grapharrowtype{1}]
  \autonodetext{Sq}{A}
  \nodetext{Ro2}{$x$}
  \area(-2,0){(-2,4)(2,4)(2,0)(-2,0)}
  \edgetext{Sq}{Te}{XXX}[
    \opaquetextfalse]
  \dirloopedge{Ro1}(-.5,.8)(.3,1.2)[
    \graphlinewidth{.01}
    \graphlinecolour{0}
    \graphlinedash{3 1}]
  \dirbow{Te}{Ro1}{.15}[
    \graphlinecolour+{.5}
    \graphlinewidth*{2}
    \grapharrowlength*{2}
    \grapharrowwidth*{.5}
    \grapharrowwidth*{3}]
\end{graph}
```



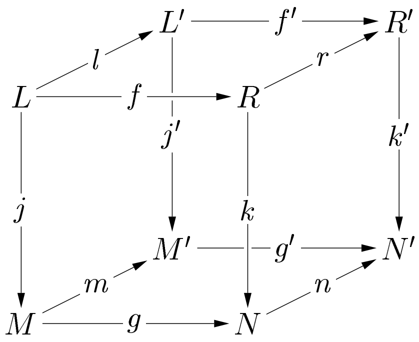
3.6 . . . und das, was von LaTeX bekannt ist:

```
\begin{graph}(4,4)(-2,0)
  \graphnodesize{.7}
  \squarenode{Sq1}(-1.5,3.5)[
    \graphnodecolour{1}]
  \squarenode{Sq2}(1.5,3.5)
  \roundnode{Ro}(0,.5)
  \diredge{Sq1}{Sq2}
  \autonodetext{Sq1}{A}
  \autonodetext{Sq2}{B}[\opaquetexttrue]
  \dirloopedge{Ro}(-.5,.8)(.3,1.2)[
    \graphlinewidth{.02}\graphlinecolour{.5}]
  \freetext(0,2){Text vom Graphen}
  \put(-1.7,3.7){\LaTeX}
  \thicklines
  \put(0,2.5){\oval(3,2)}
  \put(-2.5,1.4){\huge XXXXXXXX}
\end{graph}
```



3.7 So macht man Diagramme (zum Beispiel):

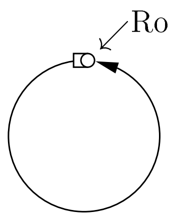
```
\begin{graph}(5,4)
  \graphlinecolour{1}\graphlinewidth{.01}
  \grapharrowlength{.2}
  \newcommand{\arr}[3]{%
    \edge{#1}{#2}[\graphlinewidth{.1}]
    \diredge{#1}{#2}[\graphlinecolour{0}]
    \edgetext{#1}{#2}{\small #3}}
  \textnode L(0,3){$L$} \textnode R(3,3){$R$}
  \textnode M(0,0){$M$} \textnode N(3,0){$N$}
  \textnode{L2}(2,4){$L'$}
  \textnode{R2}(5,4){$R'$}
  \textnode{M2}(2,1){$M'$}
  \textnode{N2}(5,1){$N'$}
  \arr{L2}{R2}{$f'$} \arr{L2}{M2}{$j'$}
  \arr{R2}{N2}{$k'$} \arr{M2}{N2}{$g'$}
  \arr L{L2}{$l$} \arr N{N2}{$n$}
  \arr R{R2}{$r$} \arr M{M2}{$m$}
  \arr LR{$f$} \arr LM{$j$}
  \arr RN{$k$} \arr MN{$g$}
\end{graph}
```



(Beachte die Definition von `\arr` und die Auswirkung auf sich schneidende Pfeile. Wer die Beschriftung *auf* den Pfeilen nicht mag, kann z.B. `\bowtext` benutzen.)

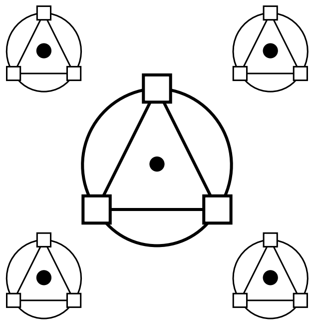
3.8 Die Reihenfolge interessiert nur bei Kommandos gleichen Typs

```
\begin{graph}(4,4)
  \autonodetext{Ro}[ne]{$\swarrow^{\mbox{Ro}}$}
  \dirbow{Sq}{Ro}{-20}
  \squarenode{Sq}(1.9,2.5)
  \roundnode{Ro}(2,2.5)
  \graphnodecolour{1}
\end{graph}
```



3.9 Zu guter Letzt: Graphen als Beschriftung

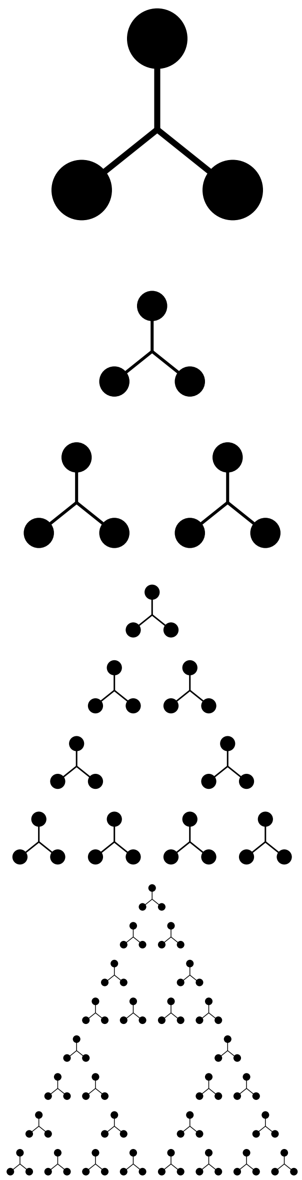
```
\begin{graph}(4,4)(-2,-2)
  \roundnode{A}(-1.5,1.5)
  \roundnode{B}(1.5,1.5)
  \roundnode{C}(1.5,-1.5)
  \roundnode{D}(-1.5,-1.5)
  \roundnode{E}(0,0)
  \autonodetext{A}{\tri{1cm}}
  \autonodetext{B}{\tri{1cm}}
  \autonodetext{C}{\tri{1cm}}
  \autonodetext{D}{\tri{1cm}}
  \autonodetext{E}{\tri{2cm}}
  \newcommand{\tri}[1]{%
    \unitlength=#1%
    \begin{graph}(0,0)
      \graphnodecolour{1}
      \squarenode{A}(0,.5)
      \squarenode{B}(-.4,-.3)
      \squarenode{C}(.4,-.3)
      \edge AB\edge BC\edge CA
      \bow CB{.3}\bow BA{.3}\bow AC{.3}
    \end{graph}%
  }
\end{graph}
```



(Beachte auch, daß Knotennamen nur lokale Bedeutung haben. So ist dies auch ein Beispiel dafür, wie man Bezeichnungen einsparen kann: Für die 20 Knoten hätten andernfalls 20 verschiedene Namen gefunden werden müssen.)

3.10 P.S.: Damit kann man sogar einfache Bildgenerierung betreiben ...

```
\newcount\depth\depth=0
\def\sierp{%
  \ifnum\depth=0%
    \terminal%
  \else%
    \nonterminal%
  \fi%
  \advance\depth by 1%
}
\def\terminal{%
  \begin{graph}(1,1)
    \roundnode{M}(.5,.45)[\graphnodesize{0}]
    \roundnode{A}(.25,.25)
    \roundnode{B}(.75,.25)
    \roundnode{C}(.5,.75)
    \edge{M}{A}
    \edge{M}{B}
    \edge{M}{C}
  \end{graph}
}
\def\nonterminal{%
  \begin{graph}(1,1)
    \advance\depth by -1
    \freetext(.25,.25){%
      \unitlength=.5\unitlength\sierp}
    \freetext(.75,.25){%
      \unitlength=.5\unitlength\sierp}
    \freetext(.5,.75){%
      \unitlength=.5\unitlength\sierp}
  \end{graph}%
}
\opaquetextfalse
\parbox{4cm}{
 \unitlength=4cm
 \sierp\\ \sierp\\ \sierp\\ \sierp
}
```



(Nett, nicht wahr? Aber zumindest dieses Beispiel läßt schon bei der nächsten Iteration (und meiner TeX-Installation) den Stack von TeX überlaufen.)

3.11 P.P.S.: Eine Variante des Vorherigen:

```
\newcount\depth\depth=0
\def\sierp{%
  \ifnum\depth=0%
    \terminal%
  \else%
    \nonterminal%
  \fi%
  \advance\depth by 1%
}
\def\terminal{%
  \begin{graph}(1,1)
    \bubble{.2}{(0,0)(1,0)(.5,1)}
  \end{graph}%
}
\def\nonterminal{%
  \begin{graph}(1,1.4)(0,-.2)
    \advance\depth by -1
    \bubble{.2}{(0,0)(1,0)(.5,1)}
    \freetext(.25,.25){%
      \unitlength=.5\unitlength\sierp}[
       \graphfillcolour+{2}]
    \freetext(.75,.25){%
      \unitlength=.5\unitlength\sierp}[
       \graphfillcolour+{2}]
    \freetext(.5,.75){%
      \unitlength=.5\unitlength\sierp}[
       \graphfillcolour+{2}]
  \end{graph}%
}
\opaquetextfalse
\parbox{4cm}{
 \unitlength=4cm
 \graphfillcolour{.4}
 \sierp\\ \sierp\\ \sierp
}
```