Cahiers GUTenberg

DE LA CONSTRUCTION DE DIAGRAMMES

Francis Borceux

Cahiers GUTenberg, nº 5 (1990), p. 41-48.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1990___5_41_0>

Cahiers GUTenberg n° 5 — Mai 1990 - spécial congrès GUTenberg'90

De la construction de diagrammes

Francis BORCEUX

Département de mathématiques UCL, 2 chemin du Cyclotron, 1348 Louvain-la-Neuve, Belgique
e-mail : FBORCEUX@BUCLLN11.bitnet

Résumé L'ensemble *diagram* de *macros* réalise la mise en page automatique de diagrammes constitués de flèches de types divers joignant des "sommets" prenant la forme d'expressions mathématiques. Le programme calcule automatiquement la longueur et la position de chaque élément. L'utilisateur peut imposer pour chaque diagramme un facteur d'échelle à sa convenance.

1. Introduction

Diverses disciplines des mathématiques font un abondant usage de "diagrammes commutatifs" constitués d'expressions mathématiques reliées par des flèches; c'est le cas notamment de l'algèbre homologique, de la théorie des catégories, de la topologie algébrique, etc...

Réaliser un diagramme simple tel que

$$\begin{array}{ccccccccc} 0 & \longrightarrow & A & \xrightarrow{f} & B & \xrightarrow{g} & C & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle\alpha} & & \downarrow{\scriptstyle\beta} & & \downarrow{\scriptstyle\gamma} & & \\ 0 & \longrightarrow & A' & \xrightarrow[f']{} & B' & \xrightarrow[g']{} & C' & \longrightarrow & 0 \end{array}$$

ne pose en soi guère de problèmes. Il suffit de s'armer d'un peu de patience et de programmer tout cela dans le jargon de l'environnement *picture* de LaTeX.

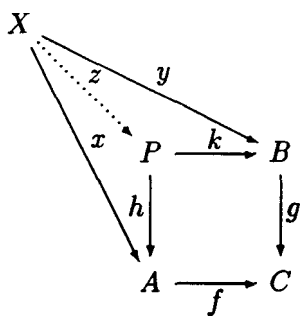
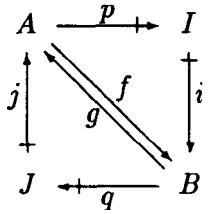


Diagramme 1

Les choses deviennent déjà moins évidentes lorqu'il s'agit de dessiner des flèches obliques, comme dans le diagramme 1, et elles se compliquent franchement lorsque pour déterminer la longueur d'une flèche, il faut tout d'abord estimer la taille qu'occuperont à l'impression les sommets du diagramme qu'elle est censée joindre:

$$\begin{array}{ccccccccc} 0 & \longrightarrow & A & \xrightarrow{f} & B & \xrightarrow{g} & C & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle\alpha} & & \downarrow{\scriptstyle\beta} & & \downarrow{\scriptstyle\gamma} & & \\ 0 & \longrightarrow & A\oplus D & \xrightarrow[f']{} & B' & \xrightarrow[g']{} & C\oplus D & \longrightarrow & 0 \end{array}$$

Et l'histoire ne s'arrête pas là! Il est fréquent que les sommets soient joints par des doubles, triples, ... voire n-uples flèches ou encore que les flèches en question soient affublées de marques conventionnelles indiquant qu'il s'agit d'une inclusion ($\hookrightarrow$), d'un quotient ($\twoheadrightarrow$) ou d'un isomorphisme ($\xrightarrow{\cong}$). Par exemple



Lorsqu'un document contient de nombreux diagrammes cumulant les diverses difficultés évoquées ci-dessus, le recours direct à l'environnement *picture* de LaTeX devient rapidement fastidieux et le fait d'arriver finalement à un résultat esthétiquement satisfaisant est souvent le fruit

d'approximations successives. La constitution d'un fichier de *macros* réalisant la composition automatique de diagrammes s'est donc avérée impérative aux yeux de plusieurs auteurs.

De nombreuses personnes ont créé leurs propres *macros* pour composer rapidement un triangle ou un rectangle en énumérant simplement comme arguments les noms des sommets et des flèches qui le composent. Mais la vraie difficulté réside bien sûr dans la mise au point d'un véritable programme capable de traiter des diagrammes de taille et de complexité arbitraires. Comme, par exemple, un "cube" du genre

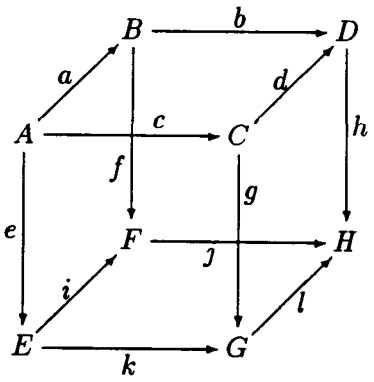


Diagramme 2

A ma connaissance, la priorité dans la mise au point de fichiers élaborés de *macros* traitant ce genre de problèmes revient à Michael BARR, professeur au département de mathématiques de l'université Mac Gill à Montréal. Partant de *macros* simples permettant de dessiner des carrés et des triangles, il regarde un diagramme complexe comme une juxtaposition de carrés et de triangles qu'il attache en divers points d'un environnement *picture*; certains *macros* facilitent cette mise en page, notamment grâce à l'introduction d'éléments "fantômes" qui n'apparaissent pas à l'impression. Le principe même sur lequel sont construits les *macros* indique à suffisance que des diagrammes un peu complexes requièrent rapidement de nombreuses manipulations élémentaires de géométrie analytique plane. Michael BARR a annoncé l'existence de sa liste *catmac* de *macros* au début '89, via un message à *TEXhax*. Des versions préliminaires de ce fichier existaient depuis quelque temps.

Le présent article décrit le fichier *diagram* qui est une collection de *macros* permettant de composer des diagrammes sur un principe radicalement différent. L'idée fondamentale est que chaque sommet ou chaque flèche n'est ni plus ni moins qu'un élément du diagramme, sans qu'aucune distinction ne soit faite entre ces divers éléments. L'utilisateur énonce les divers éléments constituant le diagramme comme il le ferait dans le cas d'une matrice et c'est le programme qui se charge d'effectuer les divers calculs de mise en page.

La version actuelle de *diagram* est opérationnelle depuis le début '89 mais aucune annonce publique à ce propos n'avait jamais été faite; le fichier se transmettait simplement "de bouche à oreille", si j'ose me permettre cette expression d'un autre âge à l'heure où les bouches et les oreilles prennent surtout la forme de terminaux reliés au courrier électronique. Le succès de ces *macros* dans le cercle toujours grandissant de ses utilisateurs fit germer l'idée que *GUTenberg'90* pourrait être l'occasion de les rendre publics. En fait un utilisateur enthousiaste a fait paraître à leur propos un avis dans *TEXhax* en janvier '90, ce qui à mon grand regret prive *GUTenberg'90* de la primeur de l'annonce.

Parmi les autres fichiers de *Macros* pour dessiner des diagrammes, je citerai celui de Paul TAYLOR (Imperial College, Londres) opérationnel depuis janvier '90.

2. Les éléments d'un diagramme

Un diagramme est donc constitué de sommets et de flèches, traités sur le même pied et considérés simplement comme les *éléments* du diagramme.

Un sommet est généralement une expression mathématique qu'il suffit de composer selon les règles habituelles. Dans l'usage du fichier *diagram*, chaque élément d'un diagramme est traité comme une formule mathématique, ce qui évite la frappe de symboles $.

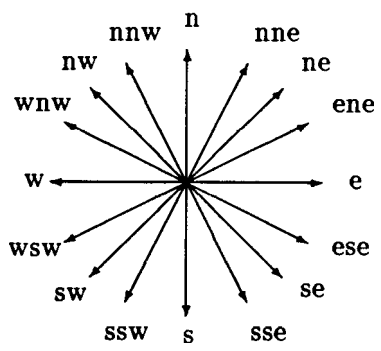
Pour ce qui est des flèches, voici les divers types disponibles et l'abbréviation (anglaise) correspondante sous laquelle le programme les reconnait.

flèche	`ar`	$\longrightarrow$
flèche pointillée	`dotar`	$\cdots\!\!\rightarrow$
monomorphisme	`mono`	$\rightarrowtail$
epimorphisme	`epi`	$\twoheadrightarrow$
bimorphisme	`bimo`	$\rightarrowtail\!\!\!\!\twoheadrightarrow$
isomorphisme	`iso`	$\xrightarrow{\cong}$
double flèche	`biar`	$\rightleftarrows$
egalité	`eql`	$=\!=$
flèches adjointes	`adjar`	$\rightleftarrows$
triple flèche	`triar`	$\rightarrow\!\!\rightarrow\!\!\rightarrow$
3 fl. adjointes	`triadjar`	$\leftarrow\!\!\rightarrow\!\!\leftarrow$
quadruple flèche	`quadriar`	$\rightarrow\!\!\rightarrow\!\!\rightarrow\!\!\rightarrow$
4 fl. adjointes	`quadriadjar`	$\leftarrow\!\!\rightarrow\!\!\leftarrow\!\!\rightarrow$
quintuple flèche	`quintiar`	$\rightarrow\!\!\rightarrow\!\!\rightarrow\!\!\rightarrow\!\!\rightarrow$
5 fl. adjointes	`quintiadjar`	$\rightarrow\!\!\leftarrow\!\!\rightarrow\!\!\leftarrow\!\!\rightarrow$

Pour insérer une flèche dans un diagramme, il faut bien entendu préciser au programme dans quelle direction il doit la tracer: horizontale, verticale, oblique, vers le haut, vers la gauche, etc... L'option choisie a été de retenir comme directions fondamentales les seize directions principales de la rose des vents, encodées via l'abbréviation anglaise correspondante. Les quatre abbréviations fondamentales sont donc

`n`	(Nord)	`s`	(Sud)
`e`	(Est)	`w`	(Ouest)

à partir desquelles on peut considérer les seize directions suivantes:



La commande décrivant une flèche est alors obtenue en faisant précéder le nom de la flèche de sa direction:

`\ndotar`	une flèche pointillée orientée vers le nord
`\swbiar`	une double flèche orientée vers le sud-ouest
`\enear`	une flèche orientée vers l'est-nord-est

etc... En fait toutes les variétés de flèches existent dans les huit directions principales de la rose des vents, mais seules les flèches ordinaires ont été programmées dans les huit directions tertiaires.

Le lecteur observateur aura constaté que notre rose des vents ferait dresser les cheveux sur la tête au plus piètre des géographes. En fait les directions tertiaires de notre rose des vents sont celles de coéfficient angulaire $\pm\frac{1}{2}$ et ± 2, c'est-à-dire les diagonales de rectangles de côtés 2 et 1. Ce sont en effet ces directions, et non celles du type $\frac{\pi}{4} + k\frac{\pi}{2}$ qui s'imposent lors d'un pavage du plan au moyen de cellules carrées. Or ce sera précisément là l'un de nos principes de base pour la construction de diagrammes.

Avant de commenter la manière dont les commandes ci-dessus ont été construites, précisons encore qu'il est possible de donner un nom à une flèche. Pour ce faire il suffit de donner comme argument à la commande le nom de la flèche en question en ayant pris soin

- de frapper en majuscule la première lettre de la commande si l'on veut imprimer le nom au-dessus de la flèche (à gauche dans le cas d'une flèche verticale);

- de frapper en majuscule la dernière lettre de la commande si l'on veut imprimer le nom en-dessous de la flèche (à droite dans le cas d'une flèche verticale).

Des règles analogues existent pour les doubles ou triples flèches, avec cette fois deux ou trois arguments. Par exemple `\Nwepi{f}` trace un epimorphisme dans la direction Nord-Ouest; cette flèche porte le nom f, imprimé au-dessus de la flèche. De même `\semonO{g}` trace un monomorphisme dans la direction Sud-Est; cette flèche porte le nom g imprimé en dessous de la flèche. Enfin `\Eadjar{f}{g}` trace deux flèches adjointes horizontales nommées f et g.

En fait, il n'y a pas grand chose à dire sur la définition des diverses commandes qui viennent d'être décrites. Relevons quand même quelques points qui auront leur importance dans la suite:

1. Chacune des commandes ci-dessus réalise le dessin escompté dans un environnement *picture* autonome de dimensions formelles (0,0); le dessin est centré par rapport à l'origine de la figure;

2. la longueur d'une flèche est laissée sous forme d'un paramètre que le programme calculera en temps utile en fonction des autres éléments du diagramme

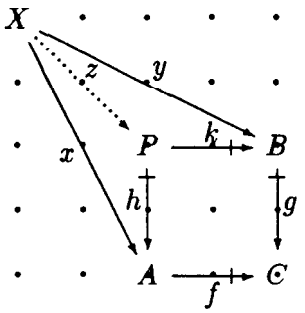
Notons encore que les noms des flèches sont automatiquement traités en mode mathématique.

3. La conception d'un diagramme

Un diagramme doit donc être pensé comme une matrice dont les divers éléments sont soit un sommet, soit une flèche, ... soit le vide! En fait la construction d'un diagramme ne se fait pas au moyen d'un environnement *array*, mais bien au moyen d'un environnement *picture*, plus adapté aux calculs un peu fins de mise en page. C'est particulièrement vrai lors d'agrandissements ou de réductions automatiques des diagrammes. Il y a cependant un prix à payer: les symboles `&` et `\\` de tabulation doivent être remplacés par des commandes ... d'où la nécessité d'inclure entre accolades les arguments de celles-ci!

Un diagramme est dès lors construit à partir d'un réseau carré formel de points, deux lignes du réseau étant séparées

par une distance laissée au choix de l'utilisateur (la valeur par défaut est 40 points). Chaque élément du diagramme est alors "attaché" en un point du réseau et centré par rapport à celui-ci. L'exemple suivant montre le diagramme 1 et le réseau de points correspondant.



La frappe de ce diagramme en taille par défaut se fait (par exemple) selon le schéma suivant

```
\DIAG
{..}                                    \nn
{..}\n{..}\n{..}                        \nn
{..}\n{..}\n{..}\n{..}\n{..}\n{..}\nn
{..}\n{..}\n{..}\n{..}\n{..}\n{..}\nn
{..}\n{..}\n{..}\n{..}\n{..}\n{..}
\diag
```

où les accolades successives contiennent les divers éléments du diagramme. La commande `\n` ("next") joue donc le rôle du symbole & dans l'environnement *array* tandis que la commande `\nn` joue le rôle du double backslash. Les deux commandes `\DIAG` et `\diag` indiquent le début et la fin de la définition du diagramme. Pour varier la taille d'un diagramme, il suffit de remplacer la première instruction par `\DIAGV{n}` et la taille globale sera multipliée par $n\%$.

On notera que les nombres de lignes et de colonnes du diagramme ne sont pas précisés. En fait les instructions `\n` et `\nn` implémentent deux compteurs comptant le nombre de lignes et de colonnes. En particulier il n'est pas nécessaire que toutes les lignes aient le même nombre d'éléments: quand une ligne ne contient plus que des éléments vides, on peut directement passer à la ligne suivante. Le compteur des colonnes retient le nombre maximum d'éléments qu'il a rencontrés dans une même ligne. Le diagramme précédent peut donc se frapper

```
\DIAG
{X}                                     \nn
{}\n{\Sedotar{z}}\n{\Esear{y}}          \nn
{}\n{\sseaR{x}}   \n{P}
  \n{\Eepi{k}}    \n {B}                \nn
{}\n{}            \n{\Smono{h}}
  \n{}            \n {\smonO{g}}\nn
{}\n{}            \n{A}
  \n{\eepI{f}}    \n {C}
\diag
```

L'instruction `\DIAG`, entre bien d'autres choses, ouvre un environnement *center* puis un environnement *picture* de dimensions formelles (0,0). C'est dans cet environnement *picture* que le diagramme est construit, le coin supérieur gauche du diagramme coincidant avec l'origine du dessin. L'instruction `\diag`, entre autres choses, clôture l'environnement *picture*, introduit un espacement horizontal et un espacement vertical calculés en fonction des nombres de colonnes et de lignes, puis ferme l'environnement *center*. C'est donc l'instruction `\diag` qui positionne le diagramme sur la page; elle peut être remplacée par l'instruction `\diagv{t}{l}{b}` qui ajoute des espacements respectifs de t points en haut du diagramme, l points à gauche et b points à droite.

La procédure précédednte de mise en page d'un diagramme implique notamment que pour insérer plusieurs petits diagrammes côte-à-côte sur une même

ligne, il convient de les considérer comme un seul grand diagramme ... dans lequel certaines "colonnes" sont entièrement vides.

4. Les calculs de mise en page

Lorsqu'ils sont introduits dans un diagramme, les divers éléments le constituant sont soit des formules mathématiques représentant un sommet, soit des flèches, soit le vide. Certains de ces éléments ont des dimensions formelles nulles: le vide, bien sûr, mais aussi les flèches; d'autres ont des dimensions non nulles, tels les sommets. Le programme ignore ces différences et même les supprime en centrant chaque élément dans une boîte de dimensions formelles (0,0) qu'il positionne au point voulu du réseau de référence.

Pour réaliser valablement le diagramme, il reste au programme à calculer la longueur la plus adéquate à donner à chacune des flèches. Ces calculs sont réalisés à partir de certaines hypothèses de travail:

- une flèche insérée en un sommet du réseau joint les deux points adjacents du réseau dans la direction où la flèche est tracée;
- un sommet du diagramme possède un nom pouvant s'étendre horizontalement d'une longueur indéfinie, mais ayant verticalement la hauteur normale d'une lettre, éventuellement affublée d'indices ou d'exposants.

Ces hypothèses de travail impliquent en particulier que la longueur d'une flèche verticale ou oblique peut être déterminée à partir des seules connaissances de sa direction et de l'écartement entre les mailles du réseau.

Le cas des flèches horizontales est beaucoup plus délicat: la longueur d'une telle flèche dépend bien sûr de l'écartement entre les points du réseau formel, mais elle doit éventuellement être adaptée en fonction des longueurs des noms des deux sommets adjacents. L'option par défaut est que toutes les flèches tracées autour d'un sommet ont leur extrémité sur un cercle de rayon 15 points autour de celui-ci. Ceci laisse 30 points pour écrire le nom d'un sommet; si un tel nom dépasse 28 points, le programme réduira la longueur des flèches horizontales adjacentes pour laisser un écart de 1 point entre le sommet et la flèche.

Une difficulté inhérente à la procédure exposée ci-avant est que la longueur à donner à une flèche ne pourra être déterminée qu'après avoir pris connaissance de l'élément suivant du diagramme. C'est la raison pour laquelle, au moment où un nouvel élément est introduit dans le diagramme,

1. cet élément est tout d'abord mesuré au moyen de la commande `\settowidth`;
2. fort de cette information, le programme calcule la longueur à donner à l'élément précédent du diagramme au cas où il s'agirait d'une flèche, puis il "imprime" cet élément;
3. enfin la définition du nouvel élément est mémorisée grâce à une instruction `\renewcommand`, en vue de son "impression" ultérieure.

Ces divers calculs requièrent l'usage de trois compteurs contenant les dimensions de l'élément précédent, de l'élément à imprimer et de l'élément suivant. Ces

diverses valeurs "glissent" d'un compteur à l'autre lors de chaque impression d'un élément. Le programme ne faisant pas la différence entre les divers éléments d'un diagramme, ces calculs sont effectués à chaque étape, mais le résultat final n'est finalement utilisé que lors de l'impression d'une flèche horizontale.

Ajoutons encore que l' agrandissement ou réduction d'un diagramme via l'instruction `\DIAGV{n}` est obtenu en modifiant l'unité de longueur des environnements *picture*; dans le cas de flèches multiples, il faut effectuer à certains endroits le changement d'échelle inverse pour éviter d'altérer l'écartement entre les flèches. Des remarques analogues concernent la taille des marques indiquant les inclusions et les quotients, ou encore l'écartement entre une flèche et son nom.

5. Quelques options

Citons en vrac quelques options permettant de réaliser des effets spéciaux dans un diagramme. Ces options ne sont en général que semi-automatisées et requièrent une certaine interaction de l'utilisateur.

Il arrive que l'on souhaite superposer deux éléments distincts en un même sommet du réseau: par exemple deux flèches, ou bien un sommet et une flèche. Cela doit se faire au moyen de la commande `\cross` qui veillera à ce que ceci ne perturbe pas les calculs de mise en page. Il faut donc frapper (par exemple)

```
\n{\cross{...}{...}}
```

Notez que cela risque d'avoir pour effet que, par exemple, le nom d'une flèche se trouve traversé par une autre flèche. Un tel défaut peut se corriger par une instruction du genre

```
\Ear{\movename{f}{4}{3}}
```

qui déplace le nom f de la flèche de 4 points vers la droite et 3 points vers le haut. Des commandes analogues `\movevertex` et `\movearrow` permettent de déplacer les sommets et les flèches d'une manière compatible avec les processus de calcul, mais il ne m'est jamais arrivé d'avoir à les utiliser. Les commandes `\cross` et `\movename` ont été utilisées dans le dessin du "cube" présenté au diagramme 2.

A côté de la commande `\cross`, une autre option parfois utile est la possibilité d'imposer au programme la longueur d'une flèche. Cela se fait en faisant suivre la commande du suffixe `v` et en donnant la longueur de la flèche comme argument. Par exemple

```
\Earv{f}{130}
```

trace une flèche de longueur 130 unités. Pour rappel, l'unité par défaut est le point, éventuellement multiplié par un facteur d'échelle via l'instruction `\DIAG{n}`. L'usage d'une telle option est impératif lorsque une flèche, fixée en un point du réseau, est appelée à joindre deux points non adjacents dans la direction où elle est tracée. Pour une flèche horizontale ou verticale, la longueur à imposer dans ce cas est donnée par la formule

$$50 + n \cdot 40$$

où n est le nombre de points "vides" du réseau que la flèche doit traverser. Dans le cas d'une flèche tracée dans une direction secondaire de la rose des vents, la formule devient

$$58 + n \cdot 40$$

Le lecteur remarquera que cette option a été utilisée pour introduire les flèches horizontales et verticales dans le dessin du "cube" présenté au diagramme 2; la

longueur imposée à ces flèches était donc de $50 + 2 \cdot 40 = 130$ points.

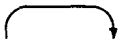
Signalons encore l'existence de flèches "courbes" introduites par une instruction du genre

```
\necurve{80}
```

ou encore

```
\Wscurve{f}{160}
```

dans le cas d'une courbe nommée f. La première flèche ci dessus prend la forme suivante, où le point indique le "centre formel" de la flèche.



De telles flèches sont généralement tracées horizontalement ou verticalement le long d'un diagramme. Le radical `curve` est précédé de deux lettres:

- la première indique la position de la flèche par rapport au diagramme;
- la seconde indique la direction générale de la flèche.

La longueur de la flèche doit être donnée comme dernier argument: ce sera (généralement) un multiple de 40. De telles flèches sont normalement introduites via une instruction `\cross`, sans qu'aucune ligne ou colonne supplémentaire ne soit ajoutée au diagramme. Le programme ne tient pas compte de leur présence dans le positionnement du diagramme sur la page; l'usage de l'instruction `\diagv` peut donc s'avérer nécessaire pour corriger l'espacement autour du diagramme.

Enfin tous les types de flèches utilisés dans les diagrammes peuvent également être insérés dans une ligne de texte, dans les deux directions horizontales. La commande pour une flèche vers la droite est simplement le radical correspondant: `\mono` pour $\longmapsto$, `\Mono{f}` pour $\xmapsto{f}$. On obtient une flèche dirigée vers la gauche en faisant préceder le radical du préfixe `bk`: `\bkbiar` pour $\rightleftharpoons$. Ces commandes peuvent indifféremment se donner en mode texte ou en mode mathématique.

D'autres options sont décrites dans le mode d'emploi des *diagram macros* (cf. [2]).

6. Mise en garde

Les *diagram macros* nécessitent la mise en mémoire de 28.295 mots et 736 commandes. Combinant cela avec LaTeX et le style *article*, on atteint 53.971 mots et 2816 commandes. Ceci peut amener à des dépassement de capacité sur certains ordinateurs personnels. Pour se tirer d'affaire, il faut donc réduire le fichier *diagram*; par exemple supprimer toutes les commandes contenant un radical peu utile à l'utilisateur: *tri*, *quadri*, *quinti*, *mono*, *epi*, *bimo*, ...

Références bibliographiques

[1] Francis Borceux, *Latex, la perfection dans le traitement de texte*, ARTEL, Bruxelles, (1989)

[2] Francis Borceux, *User's guide for the Diagram Macros*, Département de mathématiques, Louvain-la-Neuve.

[3] Donald Knuth, *The TeXbook*, Addison-Wesley.

[4] Leslie Lamport, LaTeX: *user's guide and reference manual*, Addison-Wesley.