CAHIERS GUTenberg

LUATEX 0.65 ET LES MATHÉMATIQUES

Taco HOEKWATER

Cahiers GUTenberg, nº 54-55 (2010), p. 101-127.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2010___54-55_101_0>

LuaTeX 0.65 et les mathématiques

Taco Hoekwater

Résumé. — La machinerie mathématique dans LuaTeX a été complètement modifiée depuis la version 0.40. La gestion des mathématiques dans LuaTeX a été substantiellement étendue par rapport à celle de TeX82 (donc de pdfTeX). Premièrement LuaTeX ajoute quelques primitives et en étend d'autres de telle sorte que le codage Unicode puisse facilement être utilisé en entrée. Deuxièmement, toutes les valeurs internes spéciales de TeX82 (par exemple pour l'espacement des opérateurs) sont désormais accessibles et modifiables via des séquences de contrôle. Troisièmement, des extensions permettent une utilisation plus facile des fontes mathématiques OpenType. Et finalement quelques extensions, déjà proposées dans le passé, font maintenant partie du moteur lui-même.

Abstract. — The math machinery in LuaTeX has been completely overhauled since version 0.40. The handling of mathematics in LuaTeX now extended quite a bit compared to how TeX82 (and therefore pdfTeX) handles math. First, LuaTeX adds primitives and extends some others so that Unicode input can be used easily. Second, all of TeX82's internal special values (for example for operator spacing) have been made accessible and changeable via control sequences. Third, there are extensions that make it easier to use OpenType math fonts. And finally, there are some extensions that have been proposed in the past that are now added to the engine.

This article is an update of the original article that was published in Maps 38, documenting the changes in LuaTeX between version 0.40 and version 0.65.

Note. — Traduction par Maxime Chupin de l'article de Taco Hoekwater intitulé *Math in LuaTeX 0.40*, initialement paru dans MAPS 38, 2009, p. 22-31 ; lequel a été mis à jour pour prendre en compte les changements intervenus jusqu'à la version 0.65 de LuaTeX..

Cahiers GUTenberg n° 54-55 — Octobre 2010, p. 101-127.

1. Introduction

Nous, l'équipe de développement de LuaTEX, avons commencé à réfléchir au support des maths en OpenType juste après la sortie de la fonte Cambria Math. Cependant cela nous pris plus d'un an pour obtenir l'implémentation actuelle. L'extension du moteur de composition mathématique n'est pas encore complète, mais il y a déjà assez de travail accompli pour permettre une publication. Cet article vise à donner une vision globale de tout ce qui a déjà été fait, mais cela veut aussi dire que l'on ne va pas toujours rentrer dans tous les détails. Pour une référence définitive, vous devrez lire le chapitre sur les maths dans le manuel de référence de LuaTEX [1].

2. Les primitives TEX déjà existantes

En plus des primitives mathématiques de TEX82, LuaTEX définit les primitives étendues qui ont été ajoutées par Aleph et XETEX.

2.1. TEX82

Les primitives de TEX82 ont été laissées intactes mis à part le fait que lorsqu'un caractère numérique est requis du côté gauche d'un signe d'égalité (pour `\mathcode` et `\delcode`), ce nombre peut faire usage de toute la gamme Unicode.

Exemple typique d'utilisation des primitives de TEX82 :

```
\mathcode‘\+="202B
\delcode‘\(="028300
\mathchardef\alpha="010B
\mathchar"1270
\mathaccent"017E
```

1. L'article présent utilise le format LATEX. Avec celui-ci les primitives sont à préfixer de `\luatex` comme expliqué dans l'article « Un guide pour LuaLATEX » dans ce numéro des *Cahiers*. De plus, la gestion des fontes avec les extensions `fontspec` et `unicode-math` n'est pas encore tout à fait au point, certaines des fonctionnalités présentées ci-dessous ne sont pas forcément prises en charge (c'est le cas de la fonctionnalité `ssty` qui permet de sélectionner le corps optique de façon automatique). Enfin, la seul fonte permettant d'utiliser toutes les nouveautés présentées ici est Cambria Math de Microsoft, qui peut être téléchargée facilement et qui est utilisable directement avec LuaLATEX grâce à `fontspec` et `unicode-math`. *[N.d.T.]*

```
\delimiter"3222378
\radical"270370
```

2.2. Aleph

Les primitives mathématiques du moteur Aleph utilisent une syntaxe étendue qui est dans la continuité directe de celle de TEX82. La différence est l'extension aux caractères codés sur 16 bits et à 256 familles de fontes mathématiques. Pour `\odelcode`, `\odelimiter` et `\oradical`, ceci oblige à utiliser deux entiers pour la valeur à affecter (puisque plus de 31 bits sont requis) mais à part cela, le reste de l'extension reste assez simple.

Encore une fois, LuaTEX étend à toute la gamme Unicode le code de caractère à gauche du signe d'égalité pour `\omathcode` et `\odelcode`.

Exemple typique d'utilisation des primitives d'Aleph :

```
\omathcode`\+="200002B
\odelcode`\(="000028 "030000
\omathchardef\alpha="001000B
\omathchar"1020070
\omathaccent"001007E
\odelimiter"3020022 "030078
\oradical"020070 "030070
```

2.3. XETEX

Les primitives XETEX ont besoin d'apporter encore plus d'information. En effet, comme Aleph, XETEX utilise 256 familles de fontes mathématiques, mais chacune d'entre elles utilise toute la gamme Unicode. Ceci est difficile à gérer avec une simple notation hexadécimale. C'est pourquoi les valeurs sont divisées en classe, famille, et code de caractère. Par exemple :

```
\def\overbrace {\Umathaccent 0 1 "23DE }
```

La classe est donnée par le premier entier allant de 0 à 7. L'entier suivant désigne le numéro de la famille allant de 0 à 255. Enfin, le dernier entier est le code Unicode qui s'étend de 0 à 0x10FFF en hexadécimal, ce qui correspond à 1 114 111 en décimal.

Il n'y a toujours besoin que de deux ou trois entiers puisque XETEX ne se soucie jamais de lister les versions petite et grande des délimiteurs.

L'utilisation des version grande et petite est contrôlée via les paramètres d'une fonte OpenType.

LuaTEX contient des primitives qui sont entièrement compatibles avec celles de XETEX à l'exception de leur noms, en effet, lorsque que XETEX utilise le préfixe `\XeTeX`, LuaTEX utilise `\U`.

Exemple typique d'utilisation des primitives compatibles avec celles de XETEX :

```
\Umathcode`\+="2 "0 "2B
\Udelcode`\(= "0 "28
\Umathchardef\alpha="0 "1 "B
\Umathchar "1 "2 "70
\Umathaccent "0 "1 "7E
\Udelimiter "3 "2 "22
\Uradical "2 "70
```

Pour être totalement complet, les primitives XETEX `\Umathcharnum`, `\Umathcodenum` et `\Udelcodenum` sont aussi fournies par LuaTEX mais leur utilisation est fortement déconseillée.

3. Les nouvelles extensions mathématiques

3.1. Composition « tassée » des maths

Le moteur mathématique de TEX a quatre principaux styles de composition, à savoir le style hors-texte (formule centrée), le style texte (formule au fil du texte), le style script (indices ou exposants dans une formule »texte »), et le style scriptscript (indices ou exposants de second niveau). Chacun d'entre eux, peut aussi apparaître dans une forme « tassée » qui est utilisé, par exemple, lorsque que quelque chose est en exposant d'une formule elle-même en indice (ou vice versa). Ceci fait donc un total de 8 styles de composition mathématique. Avec TEX82, il est possible de forcer un style particulier en utilisant une des primitives suivantes :

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

Cependant, jusqu'à maintenant, il n'était pas possible de passer explicitement d'un style « tassé » à un autre. Pour cela, LuaTEX ajoute les quatre nouvelles primitives que sont :

```
\crampeddisplaystyle
\crampedtextstyle
\crampedscriptstyle
\crampedscriptscriptstyle
```

3.2. Les caractères mathématiques en mode texte

LuaTEX autorise `\mathchar`, `\omathchar`, et `\Umathchar` et les séquences de contrôle qui résultent de `\mathchardef`, `\omathchardef` ou `\Umathchardef` en dehors du mode mathématique.

Lorsque LuaTEX rencontre un tel objet, il utilise la `\textfont` de la famille de maths requise pour produire un glyphe normal.

Par exemple, supposons que `\alpha` soit défini comme dans les précédents exemples et que `\omega` soit défini par `\mathchardef` à la valeur `"121`. De plus supposons que `textfont1` est `\teni` (comme dans les macros de Plain). Sous ces conditions,

```
De l'\alpha\ à l'\omega.
```

et

```
De l'{\teni\char"B} à l'{\teni \char"21}.
```

sont équivalents et produisent tous les deux :

De l'α à l'ω.

3.3. Interrogation du style courant

LuaTEX fournit une nouvelle primitive `\mathstyle` qui prend une valeur comprise entre 0 et 7 (en mode mathématique) ou -1 (dans tous les autres modes). Cette valeur représente le style courant de composition mathématique. La plus grande valeur représente le plus petit style, ainsi 0 correspond à `\displaystyle`, 1 à `\crampeddisplaystyle`, et 7 à `\crampedscriptscriptstyle`.

En utilisant ces nouvelles primitives, on peut écrire le code que voici :

```
\def\uncramped#1{{\ifcase\mathstyle
    \or \displaystyle      \or
    \or \textstyle         \or
```

```
\or \scriptstyle       \or
\or \scriptscriptstyle \fi #1}}
```

ou même créer une version entièrement développable de `\mathchoice` :

```
\def\mathchoice#1#2#3#4{{\ifcase\mathstyle
    #1\or #1\or
    #2\or #2\or
    #3\or #3\or
    #4\or #4\fi}}
```

Pour faciliter le test de la valeur de retour de `\mathstyle`, les quatre anciennes et les quatre nouvelles commandes de style mathématique ont été altérées de telle façon qu'elles peuvent être utilisées comme des valeurs numériques pour le test. Ceci permet des constructions comme ceci :

```
\ifnum\mathstyle=\textstyle
   \message{normal text style}
\fi
```

Mais il y a une piège. Il y a quelques primitives de TEX82 (`\over`, `\atop`, `\overwithdelims`, `\atopwithdelims`) pour lesquelles le style de composition qui va être utilisé n'est pas connu tant que la formule n'a pas été entièrement traitée. Ainsi, ces commandes donnent de mauvaises valeurs pour `\mathstyle`.

Pour pallier ce problème et permettre l'obtention de la bonne valeur dans tous les cas, LuaTEX fournit la nouvelle primitive `\Ustack` qui peut (ou doit) être utilisée comme préfixe pour les commandes mentionnées dans le paragraphe précédent.

```
$\Ustack{ ... a ... \over ... b ... }$
```

La commande `\Ustack` s'assure que `\mathstyle` retourne bien la bonne valeur, et ce même à l'intérieur du groupe `... a ...` . La commande `\Ustack` peut être imbriquée dans une autre si cela est nécessaire.

3.4. Les accents inférieurs

En plus du classique accent supérieur, LuaTEX supporte aussi les accents inférieurs en mode mathématique. Pour ce faire, il y a un mot clef

spécifique (`bottom`) qui peut être ajouté à `\Umathaccent`. Pour combiner les accents supérieurs et inférieurs, le mot clef est `both`. Celui-ci prend deux spécifications d'accents mathématiques. Comme toutes les nouvelles primitives qui scannent des `\mathchars` ou des `\delimiters`, celle-ci utilise une syntaxe dans la veine de celle de XeTeX[2] :

```
$$
\Umathaccent bottom "0"0"323 A
\Umathaccent both "0"0"23DE "0"0"323 A
$$
```

$$\underset{\cdot}{A}\overset{\frown}{\underset{\cdot}{A}}$$

3.5. Signes étirables horizontalement

En plus des étirements verticaux, LuaTeX gère aussi les étirements horizontaux. Ceci est particulièrement utile pour les accents mathématiques comme l'exemple suivant le montre.

```
\def\overarrow{\Umathaccent"0"0"20D7}
$$
\overarrow{a+b+c+d+e}
$$
```

$$\overrightarrow{a+b+c+d+e}$$

Notons que cette fonctionnalité dépend des paramètres de la fonte utilisée. Les exemples du présent article utilisent la fonte Cambria Math de Microsoft. Pour l'instant aucune des fontes standard de TeX ne possède ces paramètres.

4. Paramètres mathématiques

Avec LuaTeX, les métriques spécifiques qu'utilise TeX82 dans la composition mathématique sont désormais accessibles via des primitives.

2. À l'heure de l'écriture de cet article, l'extension `unicode-math` oblige à aller chercher les caractères dans la famille 4. Ainsi le code devient `\luatexUmathaccent bottom "0"4"323 A`. Cette petite modification est vraie pour le reste de l'article et est due à l'utilisation de LaTeX et `unicode-math`. *[N.d.T.]*

Ces paramètres sont initialisés pour chaque fonte mathématique à partir des métriques qu'elle fournit, ou peuvent être explicitement attribués par l'utilisateur à l'aide de commandes. Chaque paramètre existe dans huit versions correspondant aux styles de composition mathématique. La refonte du moteur de composition mathématique de LuaTEX permet de disposer de plus de paramètres que dans les moteurs précédents, même au-delà des métriques des fontes mathématiques.

4.1. Les commandes donnant accès aux paramètres mathématiques

Chacun des paramètres mathématiques peut être modifié par une commande explicite (la liste complète est donnée en table 1 en fin d'article). Voici un exemple.

```
\Umathquad\displaystyle=1em
```

Ces paramètres obéissent aux règles des groupes de TEX, mais la valeur est gelée pour une formule, et c'est la valeur active lorsque l'on atteint la fin de la formule (le dollar de fin) qui est utilisée. Voici un exemple :

```
\centerline{
 $
  \Ustack{a \over b} \times b
 $ \kern 50pt $
   \Umathfractiondenomvgap \textstyle = 8pt
   \Ustack{a \over b} \times b
 $}
```

$$\frac{a}{b} \times b \qquad \frac{a}{b} \times b$$

Vous pouvez utiliser `\the\Umathquad\displaystyle` pour récupérer la valeur courante (par exemple dans une macro de placement fin de caractères).

4.2. Les paramètres mathématiques basés sur la fonte

Alors qu'il est très intéressant de disposer de ces paramètres pour peaufiner l'apparence de certaines constructions, il serait certainement très fastidieux d'avoir à les affecter tous à la main. Pour cette raison, LuaTEX initialise (presque) tous ces paramètres à chaque fois que vous assignez une fonte à une famille de maths. Ceci se base soit sur les métriques des fontes mathématiques traditionnelles (pour les fontes

comme cmsy et cmex assignées aux familles 2 et 3, qui ont des métriques TFM), soit sur les valeurs dans la table MathConstants (lorsqu'une fonte OpenType est chargée via Lua). S'il y a une table MathConstants, les valeurs de celle-ci ont priorité sur les métriques de la fonte et, dans ce cas, peu importe à quelle famille elle est assignée. En effet, les tables MathConstants présentes dans la dernière famille assignée définissent la valeur de tous les paramètres.

Les huit paramètres sont typiquement définis en utilisant la valeur de \textfont pour les styles hors-texte et texte (« tassés » ou non), \scriptfont pour les styles script, et \sriptscriptfont pour les styles scriptscript. Ces correspondances automatiques sont recensées dans la table 2. En plus des paramètres listés dans cette table, LuaTEX prend aussi en compte la valeur de l'espace de la fonte. Pour les fontes mathématiques, celle-ci est mis à zéro.

4.3. Fractions encadrées

Il n'est pas facile de savoir comment \atopwithdelims et \overwithdelims peuvent être implémentées avec les fontes mathématiques OpenType puisque aucune des constantes dans les tables d'une telle fonte ne semble correspondre au besoin. Avec LuaTEX 0.47, nous avons décidé d'utiliser deux nouvelles valeurs de la table OpenType MathConstants : FractionDelimiterSize et FractionDelimiterDisplayStyleSize.

Ceci permet de définir les dimensions nécessaires aux primitives ...withdelims de TEX, en supposant bien sûr que la fonte a été modifiée (pendant son chargement) pour ajouter ces deux nouvelles entrée de la table OpenType.

4.4. Les paramètres d'espacement mathématique

Pour TEX82, l'espacement entre les paires de signes adjacents est contrôlé en mode mathématique par une table de huit fois huit paramètres qui est donnée dans le chapitre 18 du *TEXBook*. Avec LuaTEX, cette table a été convertie en 64 primitives de la forme \Umath...spacing, pour toutes les paires de combinaisons de bin, rel, ord, open, close, punct, inner et op.

Voici un exemple d'utilisation :

```
\centerline{$
a \times b
$ \kern 50pt $
\Umathordbinspacing \textstyle = 18mu
\Umathbinordspacing \textstyle = \thickmuskip
\thickmuskip = 10mu
a \times b
$}
```

$$a \times b \qquad\qquad a \;\times\; b$$

Normalement, on donnera à ces paramètres des dimensions en unité mathématique (`mu`). Cependant, il y a un comportement particulier lorsque qu'un des registres prédéfinis `...muskip` est utilisé. Quand l'affectation utilise `\thinmuskip`, `\medmuskip`, ou `\thickmuskip`, une modification ultérieure de ceux-ci modifiera la valeur du paramètre `\Umath...spacing` affecté (lorsque l'on affecte les `\Umath...spacing` avec des valeurs multiples de `\thinmuskip`, `\medmuskip`, ou `\thickmuskip`, les `\Umath...spacing` sont alors des pointeurs vers les valeur `\...muskip`, ce qui permet éventuellement de les modifier et que ces modifications soient prises en compte).

4.5. Version explicite des caractères de commandes

LuaTEX définit six nouvelles primitives qui ont les mêmes fonctions que ^, _, $ et $$.

Primitive	**Explication**
`\Usuperscript`	fonction de ^
`\Usubscript`	fonction de _
`\Ustartmath`	fonction de $ en dehors du mode math
`\Ustopmath`	fonction de $ en mode math
`\Ustartdisplaymath`	fonction de $$ en dehors du mode math
`\Ustopdisplaymath`	fonction de $$ en mode math

Les primitives `\Ustopmath` et `\Ustopdisplaymath` vérifient si le mode math courant est bien le bon (texte ou hors-texte), mais vous pouvez sans problème mélanger les quatre commandes de début et fin de maths avec les signes dollars.

5. Les extensions pour les maths du côté de Lua

5.1. Définir et lire les paramètres mathématiques

Les fonctions Lua `tex.setmath()` et `tex.getmath()` servent à définir ou obtenir les paramètres internes mathématiques. Voici le schéma d'utilisation :

```
tex.setmath(<string> n, <string> t, <number> n)
```

ou

```
tex.setmath('global' <string> n, <string> t, <number> n)
```

Dans un souci de simplicité d'écriture, la première chaîne de caractères est le nom du paramètre moins le préfixe « Umath », et la seconde chaîne de caractères est le nom du style moins le suffixe « style ». Par exemple :

```
tex.setmath('fractiondenomvgap','text',8*65536)
```

Un premier paramètre optionnel `global` indique qu'il s'agit d'une affectation globale. Pour le moment, vous ne pouvez pas utiliser Lua pour les paramètres d'espacement mathématiques (parce qu'il n'y a pas encore d'interface de lecture pour les distances en *unités mathématiques* « mu »).

L'obtention de la valeur d'un paramètre mathématique se fait grâce à la fonction `tex.getmath()` :

```
<number> n = tex.getmath(<string> n, <string> t)
```

laquelle suit le même schéma que la précédente.

5.2. Les attributs en mode math

Avec la venue de LuaTEX 0.40, les attributs de nœuds sont désormais retenus en mode mathématique et cela même après la conversion des maths en liste horizontale qui est ajoutée au paragraphe courant. Les nouveaux nœuds qui sont créés pendant ce processus (comme la ligne horizontale de la fraction) héritent leurs attributs du nœud parent le plus logique.

5.3. Le *callback* `mlist_to_hlist`

Un *callback* simple est accessible et peut être utilisé pour modifier au dernier moment certaines choses dans la liste de nœuds mathématique.

Lorsque vous utilisez ce *callback*, vous devez faire vous-même la conversion des maths en liste horizontale. Pour rendre cela plus simple, il existe une fonction qui réalise exactement ce que LuaTEX aurait fait s'il n'y avait pas eu d'appel au *callback*.

Commençons avec le schéma de syntaxe du *callback* :

```
function(<node> head,
         <string> displaytype,
         <boolean> need_penalties)
    return <node> newhead
end
```

Le nœud retourné doit être la tête de la liste qui doit être ajoutée à la liste verticale ou horizontale. La chaîne de caractère `displaytype` est soit `text` soit `display` en fonction de style courant. Le booléen `need_penalties` en argument est `vrai` si les pénalités doivent être insérées dans la `\hlist` générée, et `faux` sinon.

Si vous souhaitez modifier peu de choses, une approche plus simple consiste à faire les modifications en premier, et ensuite faire appel à la fonction suivante :

```
<node> h = node.mlist_to_hlist(
             <node> n,
             <string> displaytype,
             <boolean> penalties )
```

Ceci lance la conversion interne de `mlist_to_hlist` convertissant la liste mathématique `n` en la liste horizontale `h`. L'interface est exactement la même que pour le *callback* `mlist_to_hlist`, puisque que le *callback* simple est celui-ci :

```
callback.register ('mlist_to_hlist',
  function (h,d,n)
   return node.mlist_to_hlist(h,d,n)
end )
```

6. Les fonctionnalités OpenType pour les maths

Comme expliqué par Ulrik Vieth dans ses articles sur le sujet [3], les fonctionnalité mathématiques d'une fonte OpenType vont bien au-delà d'un simple support pour manipuler des caractères Unicode en entrée. Un bon nombre d'extensions ont été ajoutées à LuaTEX pour gérer des fonctionnalités spécifiques des maths OpenType.

6.1. Les métriques des fontes OpenType

Premièrement, parlons un peu des métriques de fontes OpenType. Les fontes OpenType sont chargées dans LuaTEX par du code Lua dans le *callback* `define_font` et, bien entendu, les fontes mathématiques OpenType ne font pas exception.

La fonction Lua `fontloader.to_table()` retourne les informations OpenType spécifiques aux maths en deux parties : il y a une partie globale où les constantes mathématiques sont listées, et une partie locale, par glyphe, qui contient des données telles que la correction italique et les recettes d'extension des glyphes.

La partie globale ressemble à ceci :

```
["math"]={
 ["AxisHeight"]=585,
 ...
 ["FractionDenominatorDisplayStyleGapMin"]=260,
 ["FractionDenominatorGapMin"]=133,
 ["FractionDenominatorShiftDown"]=1030,
 ["FractionNumeratorDisplayStyleGapMin"]=260,
 ["FractionNumeratorDisplayStyleShiftUp"]=1550,
 ["FractionNumeratorGapMin"]=133,
 ...
 ["ScriptPercentScaleDown"]=73,
 ["ScriptScriptPercentScaleDown"]=60,
 ...
```

La liste est bien entendu plus longue que cela : toutes les constantes mathématiques sont listées dans cette table. À part pour les quelques cas

3. Notamment « OpenType Math Illuminated », publié dans le même numéro de MAPS que le présent article. *[N.d.T.]*

dont le nom contient « Percent », les valeurs sont exprimées relativement à l'unité servant pour le tracé de la fonte (*design unit* : `u`), et celles-ci doivent être converties par du code Lua relativement à une unité TEX (*scaled point* : `sp`) avant d'être envoyées à LuaTEX (comme c'est le cas pour toutes les métriques des fontes).

L'exemple suivant est issu de Cambria Math qui est une fonte au format TrueType avec 2048 unités de dessins par cadratin, donc, si la fonte est chargée en 10pt, la valeur en *scaled points* de l'entrée « AxisHeight » sera

$$585/2048 \times 10 = 187200sp.$$

La partie locale n'est pas simple à expliquer car tous les glyphes n'ont pas le même ensemble d'informations. Pour rendre plus simple sa présentation, nous allons la diviser en deux parties. Pour cela, nous allons présenter deux exemples.

Le premier montre une partie pertinente des métriques du caractère Unicode « MATHEMATICAL ITALIC SMALL F », f :

```
{
    ["name"]="u1D453",
    ["italic_correction"]=60,
    ["mathkern"]={
     ["bottom_right"]={
      {
        ["height"]=420,
        ["kern"]=-400,
      },
      {
        ["height"]=720,
        ["kern"]=-320,
      },
      {
        ["height"]=1020,
        ["kern"]=0,
      },
     },
     ["bottom_left"]={ ... }
     ["top_right"]={ ... }
```

```
    },
    ["top_accent"]=840,
    ...
},
```

Comme vous pouvez le constater, ce glyphe a une correction italique de 60 u, une entrée `top_accent`, qui est utilisée lors du placement d'accents mathématiques, et possède aussi une sous-table `mathkern`. Celle-ci est utilisée pour les placements des indices et exposants : elle contient des paires de crénage pour chacun des quatre coins du glyphe. Chacune de ces quatre informations peut manquer dans la sous-table, ce qui signifie qu'aucune correction n'est nécessaire (c'est le cas pour le coin supérieur gauche `top_left` du glyphe présenté).

Le second exemple montre une partie des métriques pour le glyphe « SQUARE ROOT », le caractère ajustable qui représente le signe racine, √ :

```
{
 ["name"]="radical",
 ["vert_variants"]={
  ["italic_correction"]=0,
  ["parts"]={
   {
     ["component"]="uni23B7",
     ["advance"]=2743,
     ["end"]=2500,
   },
   {
     ["component"]="uni20D3",
     ["advance"]=1211,
     ["end"]=1150,
     ["extender"]=1,
     ["start"]=1150,
   },
   {
     ["component"]="radical.top",
     ["advance"]=1211,
     ["start"]=600,
```

```
     }
    },
   ["variants"]="radical radical.vsize1 \
                 radical.vsize2 radical.vsize3\
                 radical.vsize4 radical.vsize5"
  }
  ...
},
```

Ce glyphe n'a aucune information `mathkern` ni `top_accent` qui étaient présentes dans l'exemple précédent. Par contre, il contient une autre sous-table `vert_variant`. Celle-ci contient les informations pour les recettes d'extension de ce glyphe, qui sont au nombre de trois :

— La chaîne de caractère `variants` qui donne la suite des versions du glyphe de tailles croissantes,

— la table `parts` qui liste les parties extensibles,

— enfin `italic_correction` qui donne la correction italique à utiliser avec un glyphe construit à partir de ces éléments et en suivant la recette.

Les détails des métriques réelles seront expliqués plus tard.

6.2. Les corps optiques des membres des familles de fontes mathématiques OpenType

Lorsque l'on utilise des fontes mathématiques OpenType, il est important d'initialiser les corps utilisés pour `\scriptfont` et `\scriptscriptfont` selon les recommandations du créateur de la fonte (via `ScriptPercentScaleDown` et `ScriptScriptPercentScaleDown`).

Si la fonte fournit la fonctionnalité `ssty`, alors il est recommandé de l'activer (avec la valeur `ssty=1` pour le style script et `ssty=2` pour le style scriptscript). La différence par rapport aux réductions « normales » de 70% et 50% sans correction optique peut être en principe constatée sur l'exemple suivant :

x^{x^x} Cambria 7pt/5pt

x^{x^x} Cambria 73%/60%+ssty

6.3. Paramètres mathématiques supplémentaires

Les fontes mathématiques OpenType possèdent quelques paramètres de plus que les fontes mathématiques TFM traditionnelles et l'effet de ceux-ci peut être assez frappant. L'exemple ci-dessous illustre la différence en utilisant soit les valeurs par défaut de `\Umathfractiondenomvgap` et `\Umathfractionnumvgap` pour Cambria Math, soit les valeurs calculées par la règle des trois `default_rule_thickness`, qui est codée en dur dans les algorithmes de TEX82 :

$$\frac{p_p}{b^b}$$

TEX82,
règle des 3×`rule_thickness`

$$\frac{p_p}{b^b}$$

LuaTEX,
paramétrage de la fonte

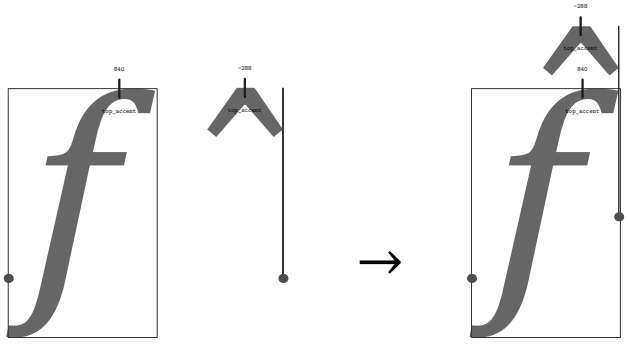
6.4. Les placements des accents mathématiques

Lorsque qu'un accent mathématique supérieur doit être placé sur un caractère qui possède un `top_accent` de valeur non nulle, cette valeur est utilisée pour positionner l'accent, et non le crénage `\skewchar` utilisé par TEX82.

La valeur `top_accent` représente une ligne verticale quelque part sur la lettre à accentuer. L'accent est donc déplacé horizontalement jusqu'à ce que sa propre ligne coïncide avec celle de la lettre de base. Si la valeur de `top_accent` de l'accent est nulle, il sera déplacé vers la droite de la moitié de sa largeur, augmentée de la correction italique.

Le placement vertical de l'accent supérieur dépend de la dimension `x_height` de la fonte contenant la lettre à accentuer (comme expliqué dans le TEXbook), mais si cette valeur se trouve être nulle, et si la fonte contient une table `MathConstants`, alors `AccentBaseHeight` est utilisé à la place. Comme une image vaut mieux qu'un long discours, voir en page suivante.

Si un accent mathématique inférieur doit être placé, c'est la valeur de `bot_accent` qui est prise en compte à la place de celle de `top_accent`. Puisque les accents inférieurs n'existent pas avec TEX82, le crénage `\skewchar` est ignoré.

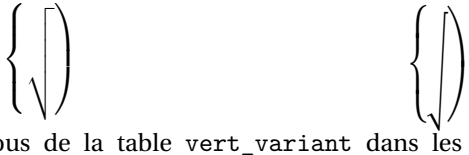


Un accent inférieur est placé juste en dessous de la lettre à accentuer sans aucune correction.

De plus, LuaTEX permet de construire des signes extensibles horizontalement, et lorsque c'est possible pour l'accent considéré et nécessité par la chasse du caractère de base, c'est utilisé par les primitives de placement d'accent pour créer des versions de largeur adaptée.

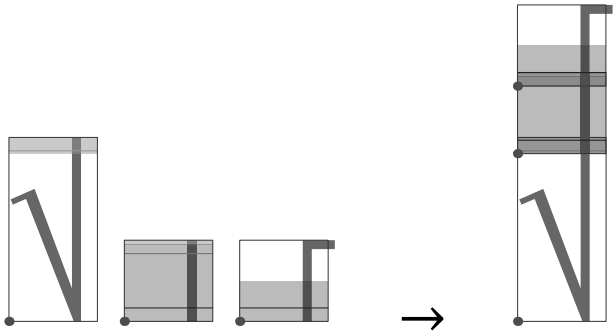
6.5. Chevauchement de signes extensibles

Dans les fontes disposant de métriques TFM, les morceaux de glyphes permettant l'extension sont placés bout à bout, ce qui donne généralement de bons résultats à l'impression, mais pose souvent des problèmes à l'écran. Si vous lisez cet article en PDF, vous verrez peut-être de petits espaces apparaître dans la partie gauche de l'exemple suivant mais pas dans la partie droite car celle-ci utilise les recettes d'extension OpenType qui introduisent un certain chevauchement lors de la construction du glyphe.



Souvenez-vous de la table `vert_variant` dans les métriques du glyphe « radivcal » donnée plus haut en page 116. Chacune des entrées `parts` possédait une clef `advance` ainsi qu'une clef `start` ou `end`. Ces deux dernières valeurs sont combinées avec la valeur `MinConnectorOverlap` de la table OpenType `MathConstants` pour définir les zones de chevauchement. Voici une illustration de processus

(l'algorithme est documenté dans la spécification OpenType Math et repris par LuaTEX).



6.6. Les grands opérateurs extensibles

Avec OpenType Math (donc avec LuaTEX), les grands opérateurs peuvent être utilisés dans plus de tailles que les deux seules fournies par TEX82. Les grands opérateurs peuvent même être construits avec des parties extensibles.

Normalement, le créateur d'une fonte OpenType décide de la taille à utiliser dans le style hors-texte (`\displaystyle`) grâce à la table `MathConstants`, mais il peut être amusant de changer cette valeur manuellement. Par exemple :

```
\Umathoperatorsize\displaystyle = 15pt
$$\sum_{k=2}^4 k^2 = 2^2 + 3^2 + 4^2 = 29$$
\Umathoperatorsize\displaystyle = 55pt
$$\sum_{k=2}^4 k^2 = 2^2 + 3^2 + 4^2 = 29$$
```

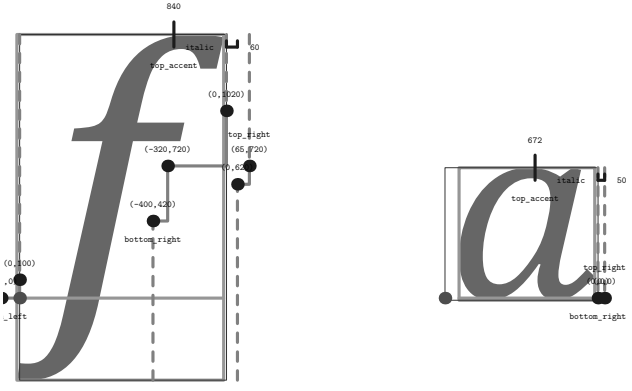
$$\sum_{k=2}^{4} k^2 = 2^2 + 3^2 + 4^2 = 29$$

$$\sum_{k=2}^{4} k^2 = 2^2 + 3^2 + 4^2 = 29$$

6.7. Le placement des indices et exposants

Comme on l'a déjà vu, les métriques d'un caractère dans une fonte mathématique OpenType peuvent contenir une table `mathkern`.

La « paire de crénage mathématique » à prendre en compte à une hauteur donnée est soit la valeur de crénage fournie pour la hauteur immédiatement supérieure, soit la plus haute pour ce caractère (s'il n'y a pas de valeur donnée pour une hauteur assez grande dans ce caractère), soit simplement zéro (si le caractère n'a aucune paire `mathkern`).



Quand un indice ou un exposant doit être placé après un signe mathématique, LuaTEX teste si l'indice ou l'exposant et l'élément en question sont tous les deux des caractères atomiques. Si tel est le cas, et si les fontes des deux éléments sont au format OpenType (et non une fonte historique pour TEX), alors LuaTEX utilisera l'algorithme OpenType Math pour placer horizontalement l'indice ou l'exposant. Ceci fonctionne comme suit :

— la position verticale de l'indice ou de l'exposant est calculée ;

— la position horizontale par défaut est juste après le caractère de base ;

— pour les exposants, la correction italique du caractère de base est ajoutée ;

— pour les exposants, deux hauteurs sont calculées : la ligne de base de l'exposant (après élévation) et le haut de la base. Pour les indices, les deux mesures sont le haut de l'indice (après avoir été abaissé), et le bas de la base ;

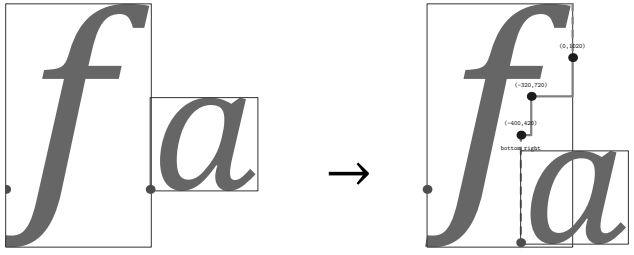
— pour chacune de ces deux hauteurs, LuaTEX

— détermine la « paire de crénage mathématique » à cette hauteur pour la base (pour un indice c'est le coin inférieur droit, pour un exposant c'est le coin supérieur droit),

— détermine la « paire de crénage mathématique » à cette hauteur pour l'indice ou l'exposant (pour un indice c'est le coin supérieur gauche, pour un exposant c'est le coin inférieur gauche) ;

— le crénage horizontal appliqué est la plus petite valeur des deux résultats de l'étape précédente.

Voici une image permettant de visualiser ceci.



6.8. Légendes pour les éléments extensibles

Les nouvelles primitives `\Uunderdelimiter` et `\Uoverdelimiter` autorisent le placement d'indice ou d'exposant sur un élément extensible, et les primitives complémentaires `\Udelimiterunder` et `\Udelimiterover` autorisent le placement d'éléments extensibles comme indices ou exposants d'un atome.

Pour ces quatre primitives, le placement vertical est contrôlé par les paramètres `\...bgap` et `\...vgap` qui sont à utiliser de la même manière que pour le placement des limites des grands opérateurs. L'exposant dans `\Uoverdelimiter` est composé dans le style script convenable et, de la même manière, l'indice dans `\Uunderdelimiter` est en style « tassé ».

```
$$
A \mathrel{\Uoverdelimiter 0 "2192 {a+b}}
B \mathrel{\Uunderdelimiter 0 "2192 {a+b}} C
$$
```

$$A \overset{a+b}{\longrightarrow} B \underset{a+b}{\longrightarrow} C$$

```
$$\Udelimiterover 0 "23DE {a+b}
+ \Udelimiterunder 0 "23DF {a+b} = C $$
```

$$\overbrace{a+b}+\underbrace{a+b}=C$$

Ici, c'est le délimiteur qui est composé en style script.

6.9. Racines avec degrés

La nouvelle primitive `\Uroot` permet la construction directe d'une racine et de son degré. La syntaxe est une simple extension de celle de `\Uradical` :

```
\Uradical <fam> <char> <radicand>
\Uroot    <fam> <char> <degree> <radicand>
```

Le placement du degré est contrôlé par le paramètres `\Umathradicaldegree...`, et le degré est composé en `\scriptscriptstyle`.

```
$$
\Uroot 0 "221A {3}{x^3+y^3}
$$
```

$$\sqrt[3]{x^3+y^3}$$

Cette partie de la spécification OpenType Math est une conversion directe de la macro plain TEX `\root \of` avec pour différence importante le transfert de la valeur *ad hoc* de placement de la macro plain à celle de la fonte, laissant ainsi le créateur de la fonte déterminer la valeur adéquate pour le rendu visuel.

Dans LuaTEX, cette fonctionnalité aurait pu être implémentée par une macro. Cependant, cela aurait était maladroit à cause de la nécessité de prendre en compte les différentes métriques.

7. Les problèmes encore ouverts

Il y a quelques problèmes qui n'ont pas encore été résolus au moment de la rédaction de cet article :

— la signification exacte de la constante `DelimitedSubFormulaMinHeight` n'est toujours pas vraiment comprise ;

— il n'est pas absolument évident de savoir quand appliquer ou non la correction italique. L'implémentation actuelle est la « meilleure estimation » obtenue à partir d'une analyse de la sortie de MS Word.

Il y a encore deux fonctionnalités des fontes OpenType Math qui n'ont pas été implémentées :

— les fractions asymétriques (en style texte) ;

— les accents aplatis pour les hauts caractères.

Ceci, comme d'autre extensions mathématiques, est planifié pour de prochaines versions de LuaTeX.

◖ Taco Hoekwater
Elvenkind
Spuiboulevard 269
3311 GP Dordrecht
Pays-Bas
taco@elvenkind.com
◖ Maxime Chupin
mc@melusine.eu.org

Table 1: Succincte description des primitives mathématiques

Nom de la primitive	Description
`\Umathquad`	Largeur de 18 mu
`\Umathaxis`	Hauteur de l'axe vertical central de la formule mathématique au-dessus de la ligne de base
`\Umathoperatorsize`	Taille minimale des grands opérateurs en mode hors-texte
`\Umathoverbarkern`	Espace libre au-dessus de la barre placée au-dessus (`\overline`)
`\Umathoverbarrule`	Largeur de la barre
`\Umathoverbarvgap`	Espace libre au-dessous de la barre
`\Umathunderbarkern`	Espace libre au-dessus de la barre placée en dessous (`\underline`)
`\Umathunderbarrule`	Largeur de la barre
`\Umathunderbarvgap`	Espace libre au-dessous de la barre
`\Umathradicalkern`	Espace libre au-dessous de la barre englobant la racine
`\Umathradicalrule`	Largeur de la barre de la racine
`\Umathradicalvgap`	Espace libre au-dessous de la barre de la racine
`\Umathradicaldegreebefore`	Décalage vers l'avant avant placement du degré de la racine
`\Umathradicaldegreeafter`	Décalage vers l'arrière après placement du degré de la racine
`\Umathradicaldegreeraise`	Pourcentage de la hauteur totale et de la profondeur du radical par lequel le degré est relevé, il est exprimé en pourcentage, ainsi l'entier 60 correspond à 60%
`\Umathstackvgap`	Espace libre vertical entre deux éléments d'une fraction encadrée`\atop`
`\Umathstacknumup`	Déplacement vers le haut du numérateur d'une fraction `\atop`
`\Umathstackdenomdown`	Déplacement vers le bas du dénominateur d'une fraction `\atop`
`\Umathfractionrule`	Largeur de la barre d'une fraction `\over`
`\Umathfractionnumvgap`	Espace libre vertical entre le numérateur et la barre de la fraction`\over`
`\Umathfractionnumup`	Déplacement vers le haut du numérateur d'une fraction `\over`
`\Umathfractiondenomvgap`	Espace libre vertical entre le dénominateur et la barre de la fraction`\over`
`\Umathfractiondenomdown`	Déplacement vers le bas du dénominateur d'une fraction `\over`
`\Umathfractiondelsize`	Taille minimale de l'encadrement pour `\...withdelims`

Nom de la primitive	Description
\Umathlimitabovevgap	Espace libre vertical pour les limites au-dessus des opérateurs
\Umathlimitabovebgap	Espace libre vertical entre la ligne de base de la limite et l'opérateur sur lequel elle se place
\Umathlimitabovekern	Espace libre vertical au-dessus de la limite
\Umathlimitbelowvgap	Espace libre vertical pour les limites au-dessous des opérateurs
\Umathlimitbelowbgap	Espace libre vertical entre la ligne de base de la limite et l'opérateur sous lequel elle se place
\Umathlimitbelowkern	Espace libre vertical au-dessous de la limite
\Umathoverdelimitervgap	Espace libre vertical pour les limites au-dessus des délimiteurs
\Umathoverdelimiterbgap	Espace libre vertical entre la ligne de base de la limite et le délimiteur sur lequel elle se place
\Umathunderdelimitervgap	Espace libre vertical pour les limites au-dessous des délimiteurs
\Umathunderdelimiterbgap	Espace libre vertical entre la ligne de base de la limite et le délimiteur sous lequel elle se place
\Umathsubshiftdrop	Déplacement vers le bas pour les boîtes et formules en indice
\Umathsubshiftdown	Déplacement vers le bas pour les caractères en indice
\Umathsupshiftdrop	Déplacement vers le haut pour les boîtes et formules en exposant
\Umathsupshiftup	Déplacement vers le haut pour les caractères en exposant
\Umathsubsupshiftdown	Déplacement vers le bas pour un indice d'un exposant
\Umathsubtopmax	Hauteur maximale que ne peuvent dépasser les exposants
\Umathsupbottommin	Hauteur minimale que ne peuvent dépasser les indices
\Umathsupsubbottommax	Distance minimale du bas de la ligne de base à laquelle doit se trouver l'indice d'une combinaison d'exposant et d'un indice
\Umathsubsupvgap	Espace libre vertical entre un indice et un exposant
\Umathspaceafterscript	Espace additionnel ajouté à la suite d'un indice ou exposant
\Umathconnectoroverlapmin	Chevauchement minimal lors de la recette d'extension d'un glyphe

Table 2: Initialisation des paramètres à partir des informations de la fonte. Dans la dernière colonne, « rule » correspond à `default_rule_thickness` et « math_x » correspond à `math_x_height`. Voir le manuel de référence de LuaTEX pour plus de détails.

Variables	**Valeur par défaut OpenType**	**Valeur par défaut TFM**
`\Umathaxis`	AxisHeight	axis_height
`\Umathoperatorsize`	MinimumDisplayOperatorHeight	<not set>
`\Umathfractiondelsize`	FractionDelimiter[DisplayStyle]Size	delim1, delim2
`\Umathfractiondenomdown`	FractionDenominator[DisplayStyle]ShiftDown	denom1, denom2
`\Umathfractiondenomvgap`	FractionDenominator[DisplayStyle]GapMin	3*rule, rule
`\Umathfractionnumup`	FractionNumerator[DisplayStyle]ShiftUp	num1, num2
`\Umathfractionnumvgap`	FractionNumerator[DisplayStyle]GapMin	3*rule, rule
`\Umathfractionrule`	FractionRuleThickness	rule
`\Umathlimitabovebgap`	UpperLimitBaselineRiseMin	big_op_spacing3
`\Umathlimitabovekern`	0	big_op_spacing5
`\Umathlimitabovevgap`	UpperLimitGapMin	big_op_spacing1
`\Umathlimitbelowbgap`	LowerLimitBaselineDropMin	big_op_spacing4
`\Umathlimitbelowkern`	0	big_op_spacing5
`\Umathlimitbelowvgap`	LowerLimitGapMin	big_op_spacing2
`\Umathoverdelimitervgap`	StretchStackGapBelowMin	big_op_spacing1
`\Umathoverdelimiterbgap`	StretchStackTopShiftUp	big_op_spacing3
`\Umathunderdelimitervgap`	StretchStackGapAboveMin	big_op_spacing2
`\Umathunderdelimiterbgap`	StretchStackBottomShiftDown	big_op_spacing4
`\Umathoverbarkern`	OverbarExtraAscender	rule
`\Umathoverbarrule`	OverbarRuleThickness	rule
`\Umathoverbarvgap`	OverbarVerticalGap	3*rule
`\Umathquad`	<font_size(f)>	math_quad
`\Umathradicalkern`	RadicalExtraAscender	rule
`\Umathradicalrule`	RadicalRuleThickness	<not set>
`\Umathradicalvgap`	Radical[DisplayStyle]VerticalGap	(rule+(abs(math_x)/4)),

Variables	Valeur par défaut OpenType	Valeur par défaut TFM
		(rule+(abs(rule)/4))
\Umathradicaldegreebefore	RadicalKernBeforeDegree	<not set>
\Umathradicaldegreeafter	RadicalKernAfterDegree	<not set>
\Umathradicaldegreeraise	RadicalDegreeBottomRaisePercent	<not set>
\Umathspaceafterscript	SpaceAfterScript	script_space
\Umathstackdenomdown	StackBottom[DisplayStyle]ShiftDown	denom1, denom2
\Umathstacknumup	StackTop[DisplayStyle]ShiftUp	num1, num3
\Umathstackvgap	Stack[DisplayStyle]GapMin	7*rule, 3*rule
\Umathsubshiftdown	SubscriptShiftDown	sub1
\Umathsubshiftdrop	SubscriptBaselineDropMin	sub_drop
\Umathsubsupshiftdown	SubscriptShiftDown[WithSuperscript]	sub2
\Umathsubtopmax	SubscriptTopMax	(abs(math_x*4)/5)
\Umathsubsupvgap	SubSuperscriptGapMin	4*rule
\Umathsupbottommin	SuperscriptBottomMin	(abs(math_x)/4)
\Umathsupshiftdrop	SuperscriptBaselineDropMax	sup_drop
\Umathsupshiftup	SuperscriptShiftUp[Cramped]	sup1, sup2, sup3
\Umathsupsubbottommax	SuperscriptBottomMaxWithSubscript	(abs(math_x*4)/5)
\Umathunderbarkern	UnderbarExtraDescender	rule
\Umathunderbarrule	UnderbarRuleThickness	rule
\Umathunderbarvgap	UnderbarVerticalGap	3*rule
\Umathconnectoroverlapmin	MinConnectorOverlap	0