$math

Contents

Introduction

This manual is not a systematic discussion about math in ConTEXt but more a collection of wrap-ups. The file also serves as testcase. The content can change over time and can also serve as a trigger for discussions on the mailing list. Content gets added sort of random. Suggestions are welcome.

We discuss high level as well as low level commands. Some of the low level commands (primitives) are wrapped in high level commands but you can of course always revert to bare TEX.

I won't go into much detail about typesetting beautiful math, for that I refer to the TEXbook.[1]

Hans Hagen
Hasselt NL

[1] The most beautiful math is not typeset by TEX anyway: just search on YouTube for "Mathematics" by Hollie McNish, the Metropole Orkest (conducted by Jules Buckley) and Martin Pyper.

1 Inputting math

1.1 Collapsing

When in text mode you enter a combination of combining accent and character, a composed character is assumed and often you then get one shape in your document. A similar feature is available in math mode. After some discussion and analysis of the potential clashes and confusion (thanks to Aditya Mahajan) we settled on a combination of methods: so called math lists entries that we entered in the character database and/or so called special sequences that are part of UNICODE. In the next tables we use `ml` for math list and `sp` for specials. Collapsing mode 1 only uses the specials, while 2 first checks the specials and then the math lists, and 3 does the reverse.

In the database you can find this (a few fields have been omitted):

```
[0x2260] = {
    adobename   = "notequal",
    category    = "sm",
    description = "NOT EQUAL TO",
    mathlist    = { 0x2F, 0x3D },
    mathspec    = {
        {
            class = "relation",
            name  = "neq",
        },
        {
            class = "relation",
            name  = "ne",
        },
    },
    specials    = { "char", 0x3D, 0x338 },
    unicodeslot = 0x2260,
}
```

and

```
[0x2261] = {
    adobename      = "equivalence",
    category       = "sm",
    description    = "IDENTICAL TO",
    mathclass      = "relation",
    mathextensible = "h",
    mathname       = "equiv",
    mathlist       = { 0x3D, 0x3D },
    unicodeslot    = 0x2261,
}
```

Here are a few examples:

	0	1 (sp)	2 (sp ml)	3 (ml sp)
`$==$`	==	= =	= =	≡
`$/=$`	/=	/=	≠	≠
`$>=$`	>=	>=	≥	≥

A complete list of collapses can be generated after loading one of the tracing modules:

```
\usemodule[math-ligatures]
```

This provides the command:

```
\showmathligatures
```

which gives:

ml	U+02016	‖	U+0007C U+0007C	––	‖	\Vert \Arrowvert \lVert \rVert \doubleverticalbar
sp	U+02026	…	U+0002E U+0002E U+0002E	. . .	…	\ldots \dots
sp	U+02033	″	U+02032 U+02032		″	\doubleprime
sp	U+02034	‴	U+02032 U+02032 U+02032		‴	\tripleprime
sp	U+02036	‶	U+02035 U+02035		‶	\reverseddoubleprime
sp	U+02037	‷	U+02035 U+02035 U+02035		‷	\reversedtripleprime
sp	U+02057	⁗	U+02032 U+02032 U+02032 U+02032		⁗	\quadrupleprime
ml	U+02190	←	U+0003C U+02212	<-	< −	\leftarrow \gets \underleftarrow \overleftarrow
ml	U+02192	→	U+02212 U+0003E	->	− >	\rightarrow \to \underrightarrow \overrightarrow
ml	U+02194	↔	U+0003C U+02212 U+0003E	<->	< − >	\leftrightarrow
sp	U+0219A	↚	U+02190 U+00338	←	↚	\nleftarrow
sp	U+0219B	↛	U+02192 U+00338	→	↛	\nrightarrow
sp	U+021AE	↮	U+02194 U+00338		↮	\nleftrightarrow
sp	U+021CD	⇍	U+021D0 U+00338		⇍	\nLeftarrow
sp	U+021CE	⇎	U+021D4 U+00338		⇎	\nLeftrightarrow
sp	U+021CF	⇏	U+021D2 U+00338		⇏	\nRightarrow
ml	U+021D0	⇐	U+0003C U+0003D U+0003D	<==	<= =	\Leftarrow
ml	U+021D2	⇒	U+0003D U+0003D U+0003E	==>	= ≽	\Rightarrow \imply
ml	U+021D4	⇔	U+0003C U+0003D U+0003D U+0003E	<==>	<= ≽	\Leftrightarrow
sp	U+02204	∄	U+02203 U+00338		∄	\nexists
sp	U+02209	∉	U+02208 U+00338		∉	\notin \nin
sp	U+0220C	∌	U+0220B U+00338		∌	\nni \nowns
sp	U+02224	∤	U+02223 U+00338		∤	\ndivides \nmid
sp	U+02226	∦	U+02225 U+00338		∦	\nparallel
sp	U+0222C	∬	U+0222B U+0222B		∬	\iint \iintop
sp	U+0222D	∭	U+0222B U+0222B U+0222B		∭	\iiint \iiintop
sp	U+0222F	∯	U+0222E U+0222E		∯	\oiint
sp	U+02230	∰	U+0222E U+0222E U+0222E		∰	\oiiint
ml	U+02237	∷	U+0003A U+0003A	: :	∷	\squaredots
ml	U+02239	−:	U+02212 U+0003A	-:	− :	\minuscolon
sp	U+02241	≁	U+0223C U+00338		≁	\nsim
sp	U+02244	≄	U+02243 U+00338		≄	\nsimeq
sp	U+02247	≇	U+02245 U+00338		≇	\approxnEq
sp	U+02249	≉	U+02248 U+00338		≉	\napprox
ml	U+02254	≔	U+0003A U+0003D	:=	≔	\colonequals
ml	U+02255	≕	U+0003D U+0003A	=:	≕	\equalscolon
sp	U+02260	≠	U+0003D U+00338	=	≠	\neq \ne

ml	U+02260	≠	U+0002F U+0003D	/=	/=	\neq \ne
ml	U+02261	≡	U+0003D U+0003D	==	= =	\equiv
sp	U+02262	≢	U+02261 U+00338		≢	\nequiv
ml	U+02262	≢	U+0002F U+0003D U+0003D	/==	/= =	\nequiv
ml	U+02264	≤	U+0003C U+0003D	<=	<=	\leq \le
ml	U+02265	≥	U+0003E U+0003D	>=	>=	\geq \ge
ml	U+0226A	≪	U+0003C U+0003C	<<	<<	\ll
ml	U+0226B	≫	U+0003E U+0003E	>>	>>	\gg
sp	U+0226D	≭	U+0224D U+00338		≭	\nasymp
ml	U+0226D	≭	U+0002F U+0224D	/	/≍	\nasymp
sp	U+0226E	≮	U+0003C U+00338	<	≮	\nless
ml	U+0226E	≮	U+0002F U+0003C	/<	/<	\nless
sp	U+0226F	≯	U+0003E U+00338	>	≯	\ngtr
ml	U+0226F	≯	U+0002F U+0003E	/>	/>	\ngtr
sp	U+02270	≰	U+02264 U+00338		≰	\nleq
ml	U+02270	≰	U+0002F U+0003C U+0003D	/<=	/<=	\nleq
sp	U+02271	≱	U+02265 U+00338		≱	\ngeq
ml	U+02271	≱	U+0002F U+0003E U+0003D	/>=	/>=	\ngeq
sp	U+02274	≴	U+02272 U+00338		≴	\nlesssim
sp	U+02275	≵	U+02273 U+00338		≵	\ngtrsim
sp	U+02278	≸	U+02276 U+00338		≸	\nlessgtr
sp	U+02279	≹	U+02277 U+00338		≹	\ngtrless
sp	U+02280	⊀	U+0227A U+00338		⊀	\nprec
sp	U+02281	⊁	U+0227B U+00338		⊁	\nsucc
sp	U+02284	⊄	U+02282 U+00338		⊄	\nsubset
sp	U+02285	⊅	U+02283 U+00338		⊅	\nsupset
sp	U+02288	⊈	U+02286 U+00338		⊈	\nsubseteq
sp	U+02289	⊉	U+02287 U+00338		⊉	\nsupseteq
sp	U+022AC	⊬	U+022A2 U+00338		⊬	\nvdash
sp	U+022AD	⊭	U+022A8 U+00338		⊭	\nvDash
sp	U+022AE	⊮	U+022A9 U+00338		⊮	\nVdash
sp	U+022AF	⊯	U+022AB U+00338		⊯	\nVDash
ml	U+022D8	⋘	U+0003C U+0003C U+0003C	<<<	<<<	\lll \llless
ml	U+022D9	⋙	U+0003E U+0003E U+0003E	>>>	>>>	\ggg \gggtr
ml	U+022DC	⋜	U+0003D U+0003C	=<	=<	\eqless
ml	U+022DD	⋝	U+0003D U+0003E	=>	=>	\eqgtr
sp	U+022E0	⋠	U+0227C U+00338		⋠	\npreccurlyeq
sp	U+022E1	⋡	U+0227D U+00338		⋡	\nsucccurlyeq
sp	U+022E2	⋢	U+02291 U+00338		⋢	\nsqsubseteq
sp	U+022E3	⋣	U+02292 U+00338		⋣	\nsqsupseteq
sp	U+022EA	⋪	U+022B2 U+00338		⋪	\ntriangleright
sp	U+022EB	⋫	U+022B3 U+00338		⋫	\ntriangleleft
sp	U+022EC	⋬	U+022B4 U+00338		⋬	\ntrianglelefteq
sp	U+022ED	⋭	U+022B5 U+00338		⋭	\ntrianglerighteq
ml	U+027F5	⟵	U+0003C U+02212 U+02212	<--	< −−	\longleftarrow
ml	U+027F6	⟶	U+02212 U+02212 U+0003E	-->	−− >	\longrightarrow
ml	U+027F7	⟷	U+0003C U+02212 U+02212 U+0003E	<-->	< −− >	\longleftrightarrow
ml	U+027F8	⟸	U+0003C U+0003D U+0003D U+0003D	<===	<= =	\Longleftarrow
ml	U+027F9	⟹	U+0003D U+0003D U+0003D U+0003E	===>	= ≽	\Longrightarrow
ml	U+027FA	⟺	U+0003C U+0003D U+0003D U+0003D U+0003E	<===>	<= ≽	\Longleftrightarrow
ml	U+02980	|||	U+0007C U+0007C U+0007C	|	|||	\tripleverticalbar
sp	U+02A0C	⨌	U+0222B U+0222B U+0222B U+0222B		⨌	\iiiint \iiiintop
sp	U+02A74	?	U+0003A U+0003A U+0003D	::=		\coloncolonequals
sp	U+02A75	= =	U+0003D U+0003D	==	= =	\eqeq
sp	U+02A76	= =	U+0003D U+0003D U+0003D	===	= =	\eqeqeq
ml	U+02A8B	⪋	U+0003C U+0003D U+0003E	<=>	<=>	\lesseqqgtr
ml	U+02A8C	⪌	U+0003E U+0003D U+0003C	>=<	>=<	\gtreqqless

1.2 Scripts

With Unicode providing math symbols and a limited set of super- and subscripts, it made sense to add yet another feature. The scripts were already supported for a long time, but at some point on the mailing list sequential scripts were mentioned. So here is an example of both (some fonts, like the one used for verbatim, don't have all symbols but you get the idea anyway):

```
\startformula
    (0), ²²: (²)  (s(²))  ¹¹ (¹)
\stopformula

\startformula
    \unstackscripts
    (0), ²²: (²)  (s(²))  ¹¹ (¹)
\stopformula
```

which renders the clueless formulas:

$$\boldsymbol{P}_{20}(0), \forall^2 x^{0+2}_{20} : \boldsymbol{P}_{20}(x^{0+2}_{20}) \Rightarrow \boldsymbol{P}_{20}(s(x^{0+2}_{20})) \vdash \forall^1 y^{0+1}_{20} \boldsymbol{P}_{20}(y^{0+1}_{20})$$

$$\boldsymbol{P}_{20}(0), \forall^2 x^{0+2}{}_{20} : \boldsymbol{P}_{20}(x^{0+2}{}_{20}) \Rightarrow \boldsymbol{P}_{20}(s(x^{0+2}{}_{20})) \vdash \forall^1 y^{0+1}{}_{20} \boldsymbol{P}_{20}(y^{0+1}{}_{20})$$

The `\unstackscripts` macro triggers the unstacking of super and subscripts.

2 Definitions

2.1 Special stackers

There are many math symbols but never enough. Here is an example of how you can roll out your own. We start out with nothing:

```
\definemathstackers
  [nosymbol]
  [voffset=\zeropoint,
   hoffset=\zeropoint,
   mathclass=ord,
   topoffset=\zeropoint,
   middlecommand=,
   color=maincolor]
```

You can now use this class of stackers:

```
\startformula
  \mathover [nosymbol] {"2217} {A}
  \mathover [nosymbol] {"2218} {A}
  \mathover [nosymbol] {"2219} {A}
\stopformula
```

This looks like this:

$$\overset{\ast}{\mathrm{A}}\overset{\circ}{\mathrm{A}}\overset{\bullet}{\mathrm{A}}$$

But we want proper math, which means an an italic nucleus, a properly placed accent, a shift of that accent matching the slope or the nucleus, so we actually need:

```
\definemathstackers
  [mysymbol]
  [voffset=-.30\mathexheight,
   hoffset=\zeropoint,
   mathclass=ord,
   topoffset=.4\mathemwidth,
   middlecommand=\mathematics,
   color=maincolor]
```

We show both over and under variants:

```
\startformula
  \mathover  [mysymbol]{"2217}       {A}
  \mathover  [mysymbol]{"2218}       {A}
  \mathover  [mysymbol]{"2219}       {A}
  \mathunder [mysymbol]       {"2217}{A}
```

```
  \mathunder [mysymbol]       {"2218}{A}
  \mathunder [mysymbol]       {"2219}{A}
  \mathdouble[mysymbol]{"2217}{"2217}{A}
  \mathdouble[mysymbol]{"2218}{"2218}{A}
  \mathdouble[mysymbol]{"2219}{"2219}{A}
\stopformula
```

So this time we get:

$$\overset{*}{A}\overset{\circ}{A}\overset{\bullet}{A}\underset{*}{A}\underset{\circ}{A}\underset{\bullet}{A}\overset{*}{\underset{*}{A}}\overset{\circ}{\underset{\circ}{A}}\overset{\bullet}{\underset{\bullet}{A}}$$

We can now redefine the ‘interiorset’ symbol to use 0x2217 instead of 0x2218:

```
\definemathover[mysymbol][interiorset]["2217]

\startformula
    \interiorset{A}^{\interiorset{A}^{\interiorset{A}}}
\stopformula
```

Of course normally you will not use color:

$$\overset{*}{A}^{\overset{*}{A}^{\overset{*}{A}}}$$

3 Vertical spacing

The low level way to input inline math in TEX is

```
$ e = mc^2 $
```

while display math can be entered like:

```
$$ e = mc^2 $$
```

The inline method is still valid, but for display math the $$ method should not be used. This has to do with the fact that we want to control spacing in a consistent way. In ConTEXt the vertical spacing model is rather stable although in MkIV the implementation is quite different. It has always been a challenge to let this mechanism work well with space round display formulas. This has to do with the fact that (in the kind of documents that we have to produce) interaction with already present spacing is somewhat tricky.

Of course much can be achieved in TEX but in ConTEXt we need to have control over the many mechanisms that can interact. Given the way TEX handles space around display math there is no real robust solution possible that gives visually consistent space in all cases so that is why we basically disable the existing spacing model. Disabling is easier in LuaTEX and recent versions of MkIV have been adapted to that.

In pure TEX what happens is this:

```
$$ x $$
```

$$x$$

A horizontal box (visualized by the thin rule on its baseline) get added which triggers a baselineskip. Then the formula is put below it. We can get rid of that box with `\noindent`:

```
\noindent $$ x $$
```

$$x$$

In addition (not shown here) vertical space is added before and after the formula and left- and rightskip on the edges. In fact typesetting display math goes like this:

- typeset the formula using display mode and wrap it in a box
- add an equation number, if possible in the same line, otherwise on a line below
- in the process center the formula using the available display width and required display indentation
- add vertical space above and below (depending also in displays being short in relation to the previous line
- at the same time also add penalties that determine the break across pages

Apart from the spacing around the formula and the equation number, typesetting is not different from:

```
\hbox {$ \displaystyle x $}
```

So this is what we will use by default in ConTEXt in order to better control spacing as spacing around math is a sensitive issue. Because math itself can have a narrow band, for instance a lone x, or relative much depth, as with y, or both depth and height as in $(1,2)$ and $x^2 + y_2$ and because a preceding line can have no or little depth and a following line little height, the visual appearance can become inconsistent. The default approach is to force consistent spacing, but when needed we can implement variants.

Spacing around display math is set up with `\setupformulas`:

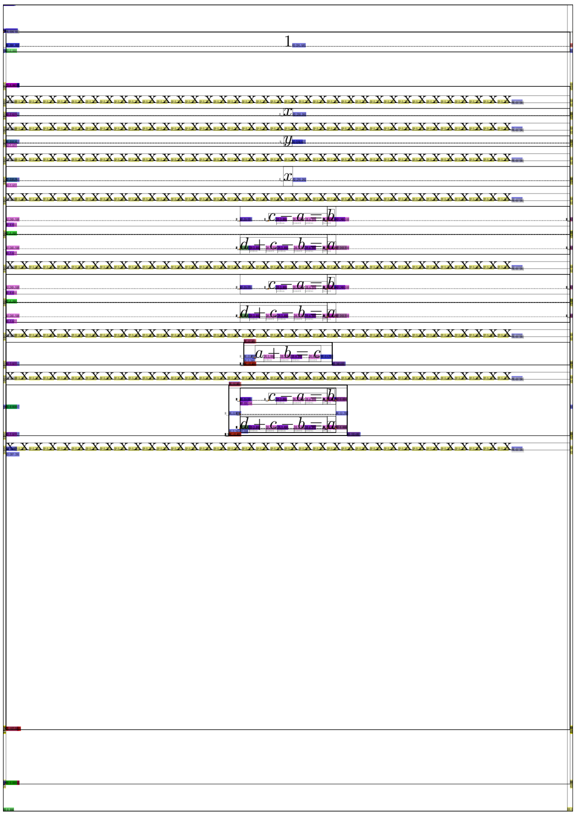
```
\setupformulas
  [spacebefore=big,
   spaceafter=big]
```

When the whitespace is larger that setting wins because as usual the larger of blanks or whitespace wins.

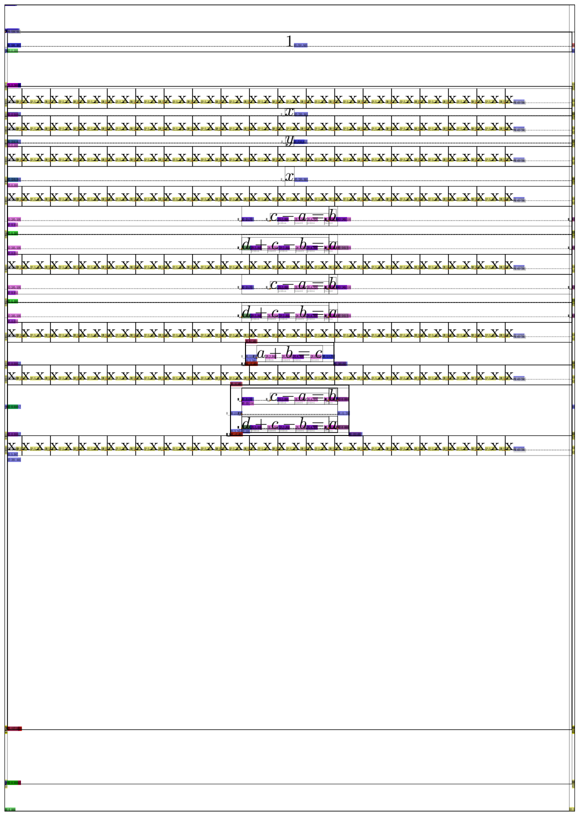
In figures 3.1, figures 3.2 and 3.3 we see how things interact. We show lines with and without maximum line height and depth (enforced by struts) alongside.

Because we want to have control over the placement of the formula number but also want to be able to align the formula with the left or right edge of the text area, we don't use the native display handler by default. We still have a way to force this, but this is only for testing purposes. By default a formula is placed centered relative to the current text, including left and right margins.

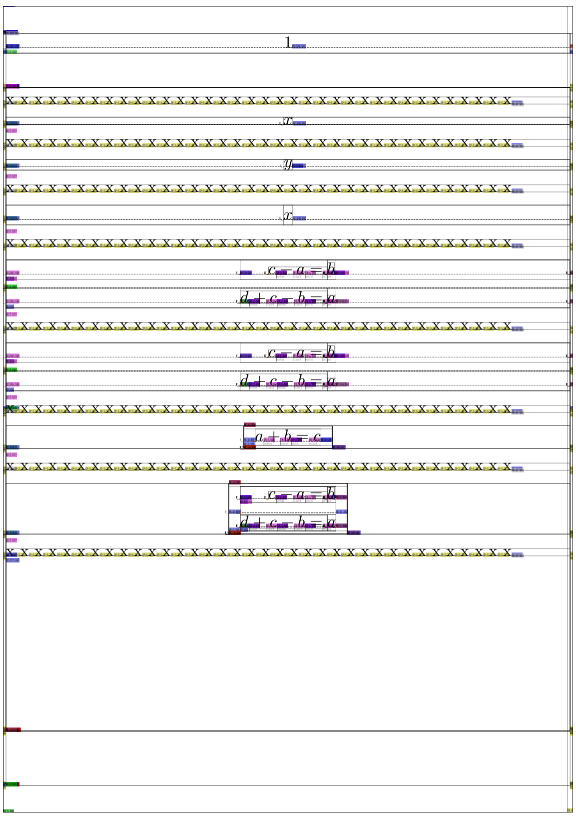
```
\fakewords{20}{40}

\startitemize
    \startitem
        \fakewords{20}{40}
        \placeformula
            \startformula
                \fakeformula
            \stopformula
    \stopitem
    \startitem
        \fakewords{20}{40}
    \stopitem
\stopitemize
```



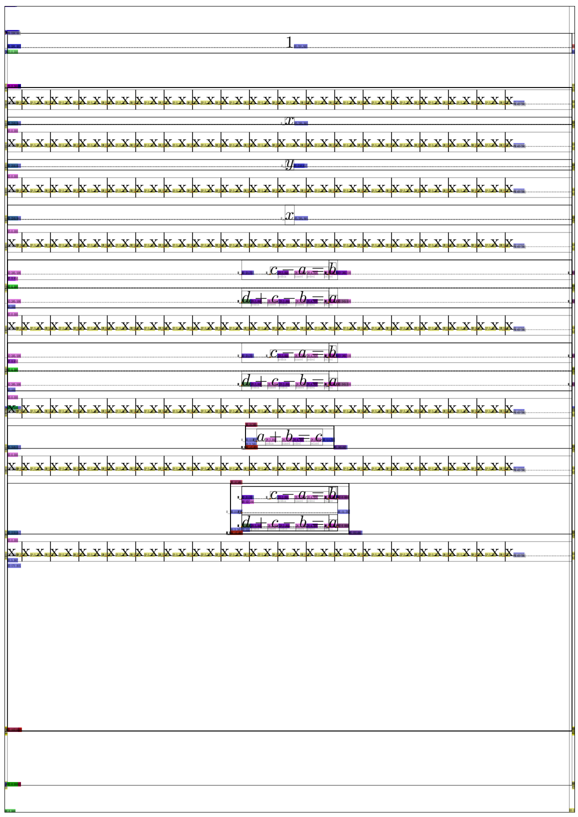
`natural + none + ws none`



`strut + none + ws none`

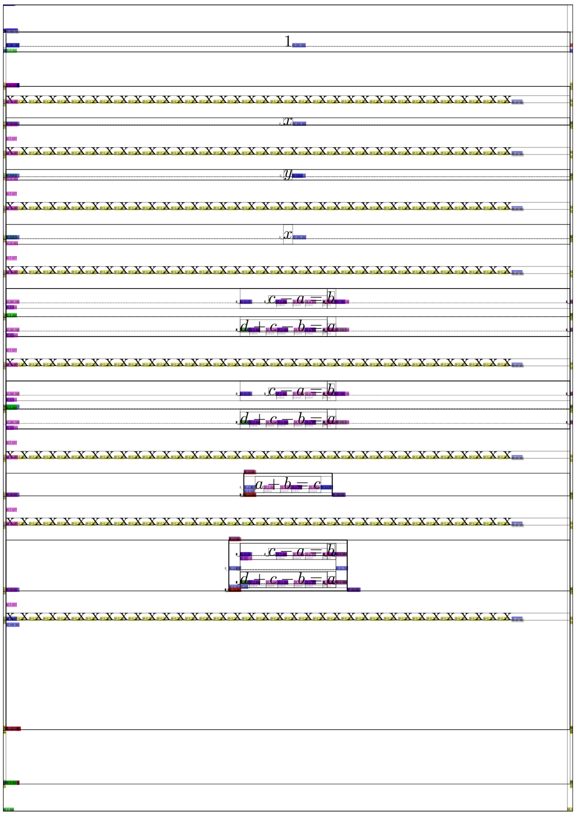


`natural + medium + ws none`

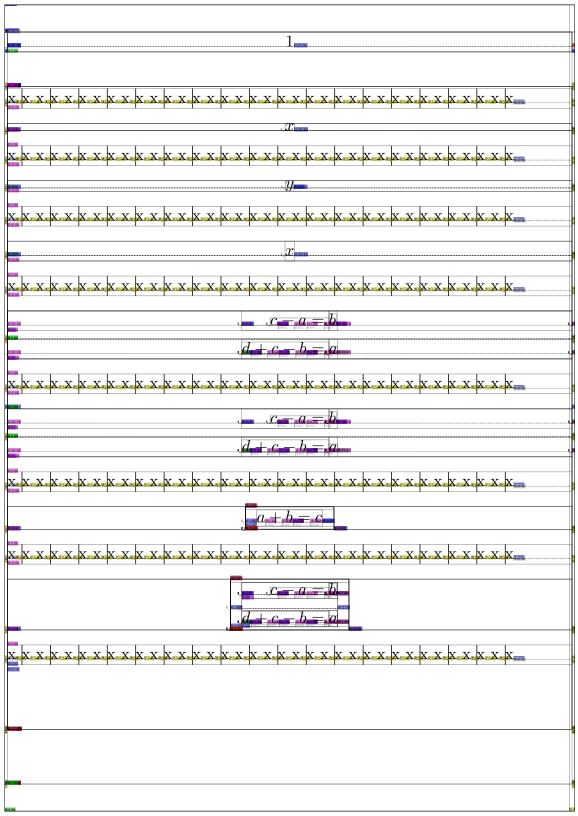


`strut + medium + ws none`

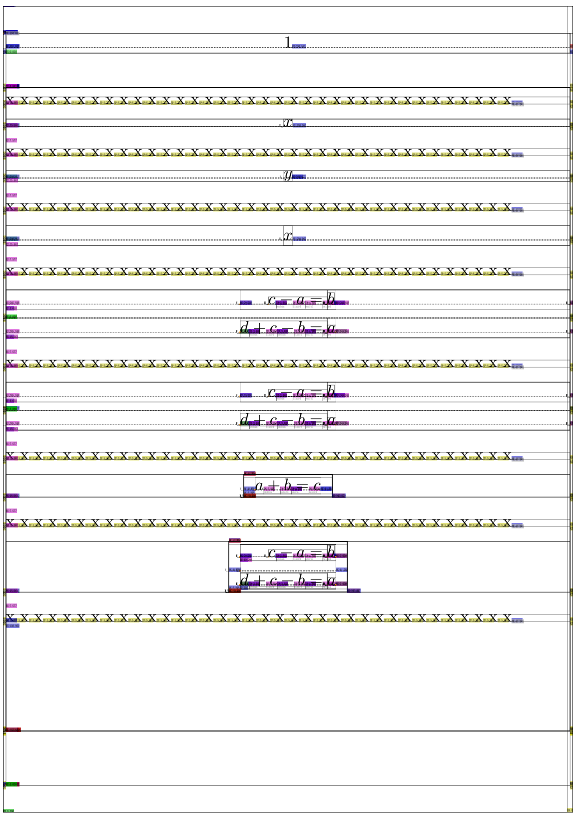
Figure 3.1 No whitespace.



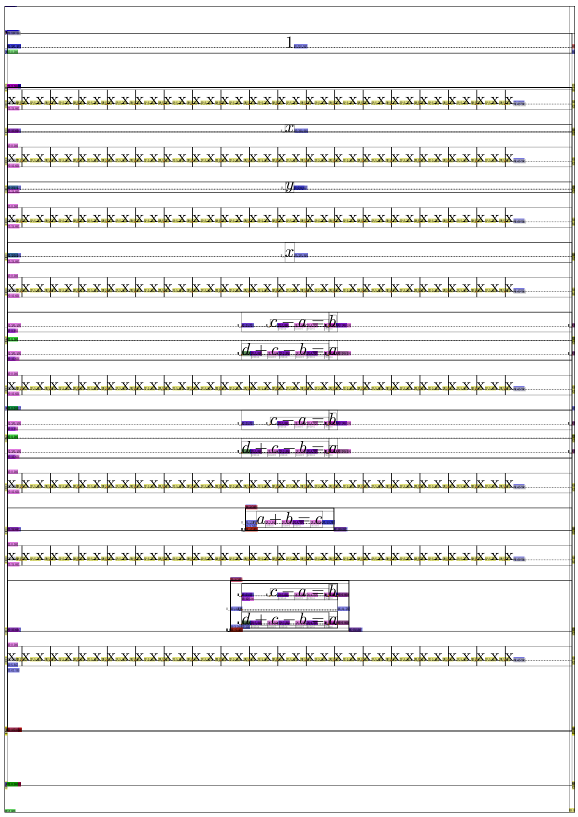
`natural + none + ws big`



`strut + none + ws big`



`natural + medium + ws big`



`strut + medium + ws big`

Figure 3.3 Whitespace larger than display spacing.

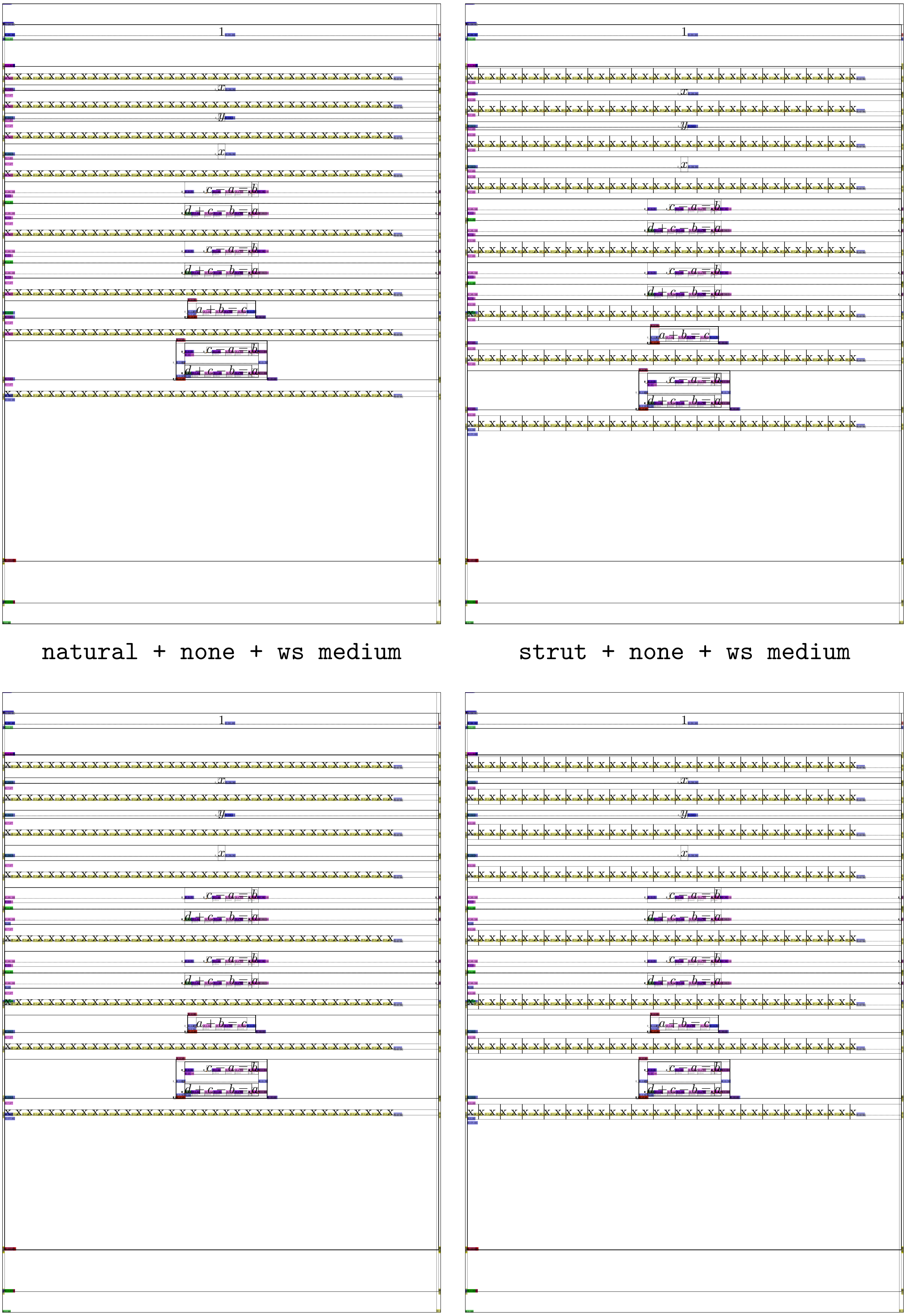


Figure 3.2 Whitespace the same as display spacing.

```
\fakewords{20}{40}\epar
```

[redacted]

- [redacted]

$$\text{[redacted]} + \text{[redacted]} + \text{[redacted]} + \text{[redacted]} = \text{[redacted]} \tag{3.1}$$

- [redacted]

[redacted]

In the next examples we explicitly align formulas to the left (`flushleft`), center (`middle`) and right (`flushright`):

```
\setupformulas[align=flushleft]
\startformula\fakeformula\stopformula
\setupformulas[align=middle]
\startformula\fakeformula\stopformula
\setupformulas[align=flushright]
\startformula\fakeformula\stopformula
```

The three cases show up as:

[redacted] + [redacted] + [redacted] + [redacted] = [redacted]

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted]

[redacted] + [redacted] + [redacted] + [redacted] = [redacted]

You can also set a left and/or right margin:

[redacted] + [redacted] + [redacted] = [redacted]

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted]

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted]

With formula numbers these formulas look as follows:

[redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.2) [redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.3)

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.4)

and the same with margins:

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.5)

[redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.6)

[redacted] + [redacted] + [redacted] + [redacted] + [redacted] = [redacted] (3.7)

When the `margin` option is set to `standard` or `yes` the current indentation (when set) or left skip is added to the left side.

```
\setupformulas[align=flushleft]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{3.8}$$

```
\setupformulas[align=flushleft,margin=standard]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{3.9}$$

The distance between the formula and the number is only applied when the formula is left or right aligned.

```
\setupformulas[align=flushright,distance=0pt]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{3.10}$$

```
\setupformulas[align=flushright,distance=2em]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{3.11}$$

3.1 Scripts

Spacing is a trade off because there is no way to predict all usage. Of course a font can be very detailed in where italic correction is to be applied and how advanced stepwise kerns are used, but not many fonts have extensive information. Here are some differences in rendering. In OpenType the super- and subscript of an integral are moved right and left half of the italic correction.

$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$
Latin Modern	Pagella	Dejavu	Cambria	Lucida OT	Xits

3.2 Bad fonts

There might be fonts out there where the italic correction is supposed to be added to the width of a glyph. In that case the following trick can be tried:

```
\definefontfeature[mathextra][italicwidths=yes] % fix latin modern
```

in which case the following might look better:

```
$\left|V\right| = \left|W\right|$
```

Of course better is to fix the font.

3.3 Multiline

Inline formulas can span lines but display math normally sits on one line unless one uses alignment mechanisms. Take this:

```
\startformula
    x\dorecurse{30}{ + #1x^{#1x}} = 10
\stopformula
```

$x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} + 13x^{13x} +$

You can set `split` to `yes` using `\setupformula` and get the following:

$$x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} +$$
$$12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x} + 21x^{21x} +$$
$$22x^{22x} + 23x^{23x} + 24x^{24x} + 25x^{25x} + 26x^{26x} + 27x^{27x} + 28x^{28x} + 29x^{29x} + 30x^{30x} = 10$$

Maybe nicer is to also set `align` to `flushleft`:

$x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} +$
$13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x} + 21x^{21x} + 22x^{22x} +$
$23x^{23x} + 24x^{24x} + 25x^{25x} + 26x^{26x} + 27x^{27x} + 28x^{28x} + 29x^{29x} + 30x^{30x} = 10$

If you want the binary operators to start the lines you can set this:

```
\setupmathematics[setups=math:spacing:split]
\setupformulas[split=yes,align=flushleft]
```

$$x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x} + 21x^{21x} + 22x^{22x} + 23x^{23x} + 24x^{24x} + 25x^{25x} + 26x^{26x} + 27x^{27x} + 28x^{28x} + 29x^{29x} + 30x^{30x} = 10$$

You can prevent a split with a large penalty. Here is a test that yuou can run to play with this feature:

```
\dostepwiserecurse {30} {100} {1} {
    \hsize \dimexpr 40pt + #1pt \relax
    \startformula
        y = a \dorecurse {50} {
            \penalty 10000 {\bf + ##1b}
             + ##1c^2
        }
    \stopformula
    \page
}
```

There is an experimental alignment mechanism available. Watch the following examples:

```
before
    \startformula
        z + 3y = \alignhere x
                 \dorecurse{20}{ + #1x^{#1x}}
    \stopformula
inbetween
    \startformula
        z + 3y \alignhere = 1
               \dorecurse{4}{
                    \dorecurse{#1}{+ #1x^{##1x}}
                    \ifnum#1<4\breakhere\fi
               }
    \stopformula
after

\setupformula
  [split=no]
```

before

$$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} +$$

inbetween

$$z + 3y = 1 + 1x^{1x} + 2x^{1x} + 2x^{2x} + 3x^{1x} + 3x^{2x} + 3x^{3x} + 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x}$$

after

```
\setupformula
  [split=yes,
   align=flushleft]
```

before

$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} +$
$12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x}$

inbetween

$z + 3y = 1 + 1x^{1x}$
$+ 2x^{1x} + 2x^{2x}$
$+ 3x^{1x} + 3x^{2x} + 3x^{3x}$
$+ 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x}$

after

```
\setupformula
  [split=yes,
   align=flushleft,
   hang=auto]
```

before

$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} +$
$11x^{11x} + 12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} +$
$20x^{20x}$

inbetween

$z + 3y = 1 + 1x^{1x}$
$+ 2x^{1x} + 2x^{2x}$
$+ 3x^{1x} + 3x^{2x} + 3x^{3x}$
$+ 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x}$

after

```
\setupformula
  [split=yes,
   align=flushleft,
   hang=auto,
   distance=1em]
```

before

$$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x}$$

inbetween

$$\begin{aligned} z + 3y = {} & 1 + 1x^{1x} \\ & + 2x^{1x} + 2x^{2x} \\ & + 3x^{1x} + 3x^{2x} + 3x^{3x} \\ & + 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x} \end{aligned}$$

after

```
\setupformula
  [split=yes,
   align=flushleft,
   hang=yes,
   distance=2em]
```

before

$$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x}$$

inbetween

$$\begin{aligned} z + 3y = {} & 1 + 1x^{1x} \\ & + 2x^{1x} + 2x^{2x} \\ & + 3x^{1x} + 3x^{2x} + 3x^{3x} \\ & + 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x} \end{aligned}$$

after

```
\setupformula
  [split=yes,
   align=flushleft,
   hang=yes,
   distance=2em,
   interlinespace=1.5\lineheight]
```

before

$$z + 3y = x + 1x^{1x} + 2x^{2x} + 3x^{3x} + 4x^{4x} + 5x^{5x} + 6x^{6x} + 7x^{7x} + 8x^{8x} + 9x^{9x} + 10x^{10x} + 11x^{11x} + 12x^{12x} + 13x^{13x} + 14x^{14x} + 15x^{15x} + 16x^{16x} + 17x^{17x} + 18x^{18x} + 19x^{19x} + 20x^{20x}$$

inbetween

$$z + 3y = 1 + 1x^{1x}$$
$$+ 2x^{1x} + 2x^{2x}$$
$$+ 3x^{1x} + 3x^{2x} + 3x^{3x}$$
$$+ 4x^{1x} + 4x^{2x} + 4x^{3x} + 4x^{4x}$$

after

If you want to split over pages, you can say:

```
\setupformula
  [split=page,
   align=middle]
```

but that is rather experimental (especially in combination with other number placement related options).

3.4 Scripts

Superscripts and subscripts are typeset in a smaller size than their nucleus. You can influence that as follows:

```
\startformula
x^{2} = x^{\textstyle 2}
      = x^{\scriptstyle 2}
      = x^{\scriptscriptstyle 2}
\stopformula
```

$$x^2 = x^{\textstyle 2} = x^{\scriptstyle 2} = x^{\scriptscriptstyle 2}$$

You can also use macros instead of a `^` and `_`, as in:

```
\startformula
x \superscript   {2} =
x \superscript   {\textstyle 2} =
x \superscript   {\scriptstyle 2} =
x \superscript   {\scriptscriptstyle 2} =
x \nosuperscript {2}
\stopformula
```

$$x^2 = x^{\textstyle 2} = x^{\scriptstyle 2} = x^{\scriptscriptstyle 2} = x^{\textstyle 2}$$

The `\nosuperscript` primitive makes sure that we get the same size as the nucleus.

```
\startformula
x \superscript   {2} \subscript   {i} =
x \nosuperscript {2} \subscript   {i} =
x \superscript   {2} \nosubscript {i} =
x \nosuperscript {2} \nosubscript {i}
\stopformula
```

$$x_i^2 = x_i^2 = x_i^2 = x_i^2$$

3.5 Text accents

You can put an accent over a character:

```
$\grave{x} \neq \grave{i}$\quad
$\ddot {x} \neq \ddot {i}$\quad
$\bar  {x} \neq \bar  {i}$\quad
$\acute{x} \neq \acute{i}$\quad
$\hat  {x} \neq \hat  {i}$\quad
$\check{x} \neq \check{i}$\quad
$\breve{x} \neq \breve{i}$\quad
$\dot  {x} \neq \dot  {i}$\quad
$\ring {x} \neq \ring {i}$\quad
$\tilde{x} \neq \tilde{i}$\quad
$\dddot{x} \neq \dddot{i}$\quad
```

This comes out as: $\grave{x} \neq \grave{i}$ $\ddot{x} \neq \ddot{i}$ $\bar{x} \neq \bar{i}$ $\acute{x} \neq \acute{i}$ $\hat{x} \neq \hat{i}$ $\check{x} \neq \check{i}$ $\breve{x} \neq \breve{i}$ $\dot{x} \neq \dot{i}$ $\mathring{x} \neq \mathring{i}$ $\tilde{x} \neq \tilde{i}$ $\dddot{x} \neq \dddot{i}$. For regular text you can better use proper composed UTF encoded characters.

3.6 Directions

Math has its own direction control:

```
\startcombination[nx=4,ny=2,distance=1cm]
    {\MathTest{0}{0}{0}} {\MathShow1{0}{0}{0}}
    {\MathTest{0}{0}{1}} {\MathShow2{0}{0}{1}}
    {\MathTest{0}{1}{0}} {\MathShow3{0}{1}{0}}
    {\MathTest{0}{1}{1}} {\MathShow4{0}{1}{1}}
    {\MathTest{1}{0}{0}} {\MathShow5{1}{0}{0}}
    {\MathTest{1}{0}{1}} {\MathShow6{1}{0}{1}}
    {\MathTest{1}{1}{0}} {\MathShow7{1}{1}{0}}
    {\MathTest{1}{1}{1}} {\MathShow8{1}{1}{1}}
\stopcombination
```

Normally you will not control directions this way but use the proper parameters in layout related setup commands.

$a^2 + b^2 = c^2$	$a^2 + b^2 = c^2$	$a^2 + b^2 = c^2$	$a^2 + b^2 = c^2$
1 : m=0 t=0 p=0	2 : m=0 t=0 p=1	3 : m=0 t=1 p=0	4 : m=0 t=1 p=1
${}^2c = {}^2b + {}^2a$	${}^2c = {}^2b + {}^2a$	${}^2c = {}^2b + {}^2a$	${}^2c = {}^2b + {}^2a$
5 : m=1 t=0 p=0	6 : m=1 t=0 p=1	7 : m=1 t=1 p=0	8 : m=1 t=1 p=1

3.7 Surround

The spacing around inline formulas is consistent with other spacing but it can be enlarged. We just show a few examples:

```
\hsize 20em
We have
\dorecurse {8} {%
    \ifcase#1\or\else and \fi
    $x+#1$ and $x-#1$ and $x \times #1$
}
\removeunwantedspaces .
\par
```

We have $x + 1$ and $x - 1$ and $x \times 1$ and $x + 2$ and $x - 2$ and $x \times 2$ and $x + 3$ and $x - 3$ and $x \times 3$ and $x + 4$ and $x - 4$ and $x \times 4$ and $x + 5$ and $x - 5$ and $x \times 5$ and $x + 6$ and $x - 6$ and $x \times 6$ and $x + 7$ and $x - 7$ and $x \times 7$ and $x + 8$ and $x - 8$ and $x \times 8$.

```
\setupmathematics
  [textdistance=2pt plus 1pt minus 1pt]
```

We have $x + 1$ and $x - 1$ and $x \times 1$ and $x + 2$ and $x - 2$ and $x \times 2$ and $x + 3$ and $x - 3$ and $x \times 3$ and $x + 4$ and $x - 4$ and $x \times 4$ and $x + 5$ and $x - 5$ and $x \times 5$ and $x + 6$ and $x - 6$ and $x \times 6$ and $x + 7$ and $x - 7$ and $x \times 7$ and $x + 8$ and $x - 8$ and $x \times 8$.

```
\setupmathematics
  [textdistance=4pt plus 2pt minus 2pt]
```

We have $x + 1$ and $x - 1$ and $x \times 1$ and $x + 2$ and $x - 2$ and $x \times 2$ and $x + 3$ and $x - 3$ and $x \times 3$ and $x + 4$ and $x - 4$ and $x \times 4$ and $x + 5$ and $x - 5$ and $x \times 5$ and $x + 6$ and $x - 6$ and $x \times 6$ and $x + 7$ and $x - 7$ and $x \times 7$ and $x + 8$ and $x - 8$ and $x \times 8$.

3.8 Choices

The next examples are generated using this macro:

```
\starttexdefinition unexpanded Test#1#2

    \begingroup

    \showmakeup[depth]

    \def\TestA{\dontleavehmode\ruledhbox{\dorecurse{8}{before }}}
    \def\TestB{\dontleavehmode\ruledhbox{\dorecurse{8}{after }}}
    \def\TestC{\dorecurse{18}{x+}x}

    \setdisplaymathspacemodel[3]
    \setupalign[flushleft]  1\space:\space\TestA \par
    \startformula #2 \TestC \stopformula \par
    \setupalign[flushleft]  2\space:\space\TestB \par

    \setdisplaymathspacemodel[4]

    \vskip#1\lineheight

    \setupalign[flushright] \TestA\space:\space2 \par
    \startformula #2 \TestC \stopformula \par
    \setupalign[flushright] \TestB\space:\space2 \par

    \endgroup

\stoptexdefinition
```

It demonstrates the often hard decisions that we have to make with regards to spacing. On the one hand we want to be adaptive, on the other hand we want to be consistent, for instance in the depth of lines. These examples overlay the two variants (which is of course font and style dependent).

1 : before before before before before before before before

before before before before before before before before : 2

$$x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x$$

2 : after after after after after after after after

after after after after after after after after : 2

1 : before before before before before before before before

before before before before before before before before : 2

$$\frac{1}{2}x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x$$

2 : after after after after after after after after

after after after after after after after after : 2

1 : before before before before before before before before

before before before before before before before before : 2

$$\frac{1}{\frac{1}{2}}x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x + x$$

2 : after after after after after after after after

after after after after after after after after : 2

One side effect of these options is that at some point we need to choose a default and then easily forget about the other variants.

4 Framing

The `\framed` macro is one of the core constructors in ConTEXt and it's used all over the place. This macro is unlikely to change its behaviour and as it has evolved over years it comes with quite some options and some can interfere with the expectations one has. In general using this macro works out well but you need to keep an eye on using struts and alignment.

```
\framed{$e=mc^2$}
```

The outcome of this is:

$e = mc^2$

There is a bit of offset (that you can set) but also struts are added as can be seen when we visualize them:

$e = mc^2$

These struts can be disabled:

```
\framed[strut=no]{$e=mc^2$}
```

Now the result is more tight.

$e = mc^2$

These struts are the way to get a consistent look and feel and are used frequently in ConTEXt. We mention these struts because they get in the way when we frame a display formula. Let's first look at what happens when we just package a formula in a box:

```
\vbox\bgroup
    \startformula
        e = mc^2
    \stopformula
\egroup
```

We get:

$$e = mc^2$$

Now there are a few properties of displaymath that one needs to keep in mind when messing around with them this way. First of all display math is meant to be used as part of the page stream. This means that spacing above and below is adapted to what comes before and after. It also means that, because formulas can be numbered, we have some settings that relate to horizontal placement.

The default vertical spacing is easy to get rid of:

```
\vbox\bgroup
    \startformula[packed]
        e = mc^2
    \stopformula
\egroup
```

This gives:

$$e = mc^2$$

Another handy keyword is `tight`:

```
\vbox\bgroup
    \startformula[tight]
        e = mc^2
    \stopformula
\egroup
```

This gives:

$$e = mc^2$$

We can combine these two:

```
\vbox\bgroup
    \startformula[packed,tight]
        e = mc^2
    \stopformula
\egroup
```

This gives:

$$e = mc^2$$

Just in case you wonder why we need to go through these troubles: keep in mind that we are wrapping something (math) that normally goes in a vertical list with text above and below.

The `packed` and `tight` options can help when we want to wrap a formula in a frame:

```
\framed
    [strut=no]
    {
        \startformula[packed,tight]
            e = mc^2
        \stopformula
```

```
    }
```

which renders as:

$\boxed{e = mc^2}$

There is a dedicated math framed instance that is tuned to give better results and automatically switches to math mode:

```
\mframed {
    e = mc^2
}
```

becomes:

$\boxed{e = mc^2}$

Framing a formula is also supported as a option, where the full power of framed can be applied to the formula. We will illustrate this in detail on the next pages. For this we use the following sample:

```
% language=us runpath=texruns:manuals/math

\setuplayout[topspace=5mm,bottomspace=5mm,height=middle,header=1cm,footer=0cm]

\starttext

\startbuffer[sample]
    \enabletrackers[formulas.framed] \showboxes
    \startformula
        e = mc^2
    \stopformula
    \par
    \startformula
        e = mc^2
    \stopformula
    \startformula
        e = mc^2
    \stopformula
    \startformula
        e \dorecurse{12} { = mc^2 }
    \stopformula
    \startplaceformula
        \startformula
            e = mc^2
```

```
        \stopformula
    \stopplaceformula
    \startplaceformula
        \startformula
            e \dorecurse{12} { = mc^2 }
        \stopformula
    \stopplaceformula
\stopbuffer

\startbuffer[setup-b]
\setupformula
  [option=frame]
\stopbuffer

\startbuffer[setup-d]
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]
\stopbuffer

\startbuffer[setup-c]
\setupformula
  [frame=number]
\stopbuffer

\startbuffer[all]
\start
    \typebuffer[setup-a]
    \getbuffer[setup-a]
    \getbuffer[sample]
    \typebuffer[setup-b]
    \typebuffer[setup-d]
    \getbuffer[setup-b]
    \getbuffer[setup-d]
    \getbuffer[sample]
    \typebuffer[setup-c]
    \getbuffer[setup-c]
    \getbuffer[sample]
    \page
```

```
\stop
\stopbuffer

\startbuffer
    \startbuffer[setup-a]
    \setupformula
      [align=flushleft]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
      [align=flushleft,location=left]
    \stopbuffer
    \getbuffer[all]

    \startbuffer[setup-a]
    \setupformula
      [align=middle]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
      [align=middle,location=left]
    \stopbuffer
    \getbuffer[all]

    \startbuffer[setup-a]
    \setupformula
      [align=flushright]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
      [align=flushright,location=left]
    \stopbuffer
    \getbuffer[all]
\stopbuffer

\getbuffer

\startbuffer[setup-b]
\setupformula
  [option={tight,frame}]
```

```
\stopbuffer

\getbuffer

\stoptext
```

In figure 4.1, 4.2 and 4.3 you see some combinations. You can run this example on your machine and see the details.

With each formula class a framed variants is automatically created:

```
\defineformula
  [foo]

\setupformulaframed
  [foo]
  [frame=on,
   framecolor=red]

\startfooformula[frame]
    e=mc^2
\stopfooformula
```

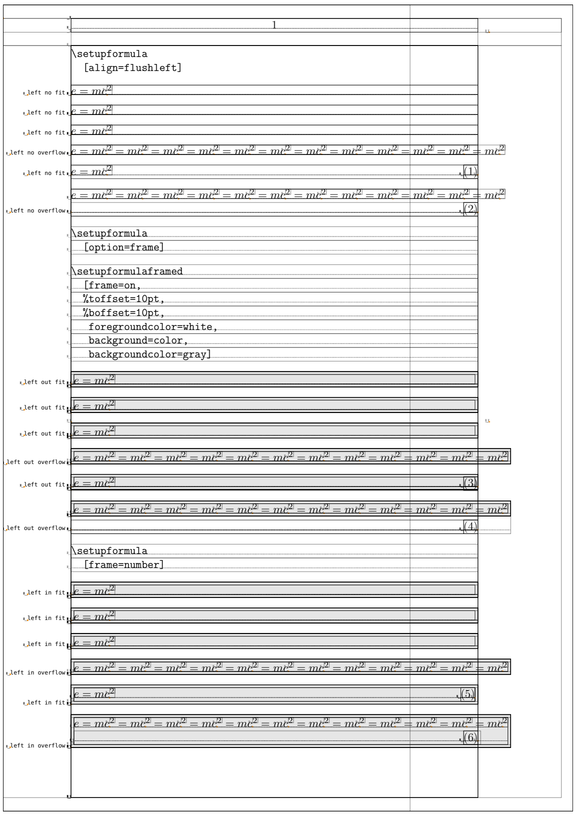
This time you get a red frame:

$$e = mc^2$$

You can also frame the number, as in:

```
\setupformulaframed[framecolor=red,frame=on,offset=1ex]
\setupformula[option=frame,color=blue]
\setupformula[numbercommand={\inframed[framecolor=green]}]

\startplaceformula
    \startformula
        2 + 2 = 2x
    \stopformula
\stopplaceformula
```

The boxes get properly aligned:

$$2 + 2 = 2x \qquad (4.1)$$

right + flushleft

right + flushleft

left + flushleft + tight

left + flushleft + tight

Figure 4.1 Framed formulas flushed left.

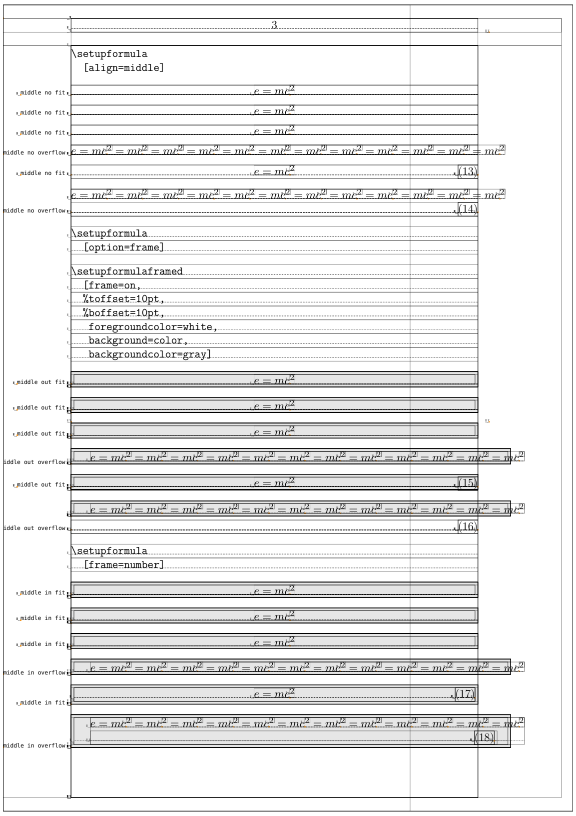
right + flushright

right + flushright

left + flushright + tight

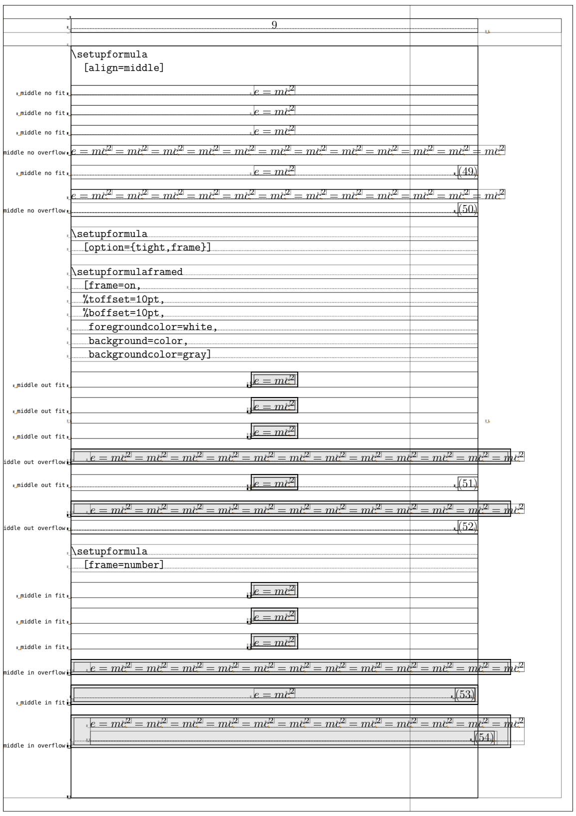
left + flushright + tight

Figure 4.3 Framed formulas flushed right.



right + middle

right + middle



left + middle + tight

left + middle + tight

Figure 4.2 Framed formulas centered.

5 Numbering

Numbering equations can be a bit of a mess. Formuals can be unnumbers, numbered, numbered with an associated reference. Numbers can go on the while formula and on the rows in an alignment. Combine that with positioning left or right and left or righ taligned formulas and the picture gets complicated. When something turns out wrong, just let me know and the respective branch in the code can be adapted. Here are some examples:

```
\startplaceformula[a]
    \startformula
        (1)
    \stopformula
\stopplaceformula
```

$$(1) \tag{5.1}$$

```
\startplaceformula[b]
    \startformula
        \startalignment
            \NC 1 \NC =     \NR
            \NC 2 \NC = (2) \NR
            \NC 3 \NC =     \NR
        \stopalignment
    \stopformula
\stopplaceformula
```

$$\begin{aligned} 1 &= \\ 2 &= (2) \\ 3 &= \end{aligned} \tag{5.2}$$

```
\startplaceformula[c]
    \startformula
        \startalignment
            \NC 1 \NC = (3) \NR[x]
            \NC 2 \NC =     \NR
            \NC 3 \NC = (4) \NR[y]
        \stopalignment
    \stopformula
\stopplaceformula
```

$$\begin{aligned} 1 &= (3) && (5.3) \\ 2 &= \\ 3 &= (4) && (5.4) \end{aligned}$$

```
\startplaceformula[d]
```

```
    \startformula
        (5)
    \stopformula
\stopplaceformula
```

$$(5) \tag{5.5}$$

```
\startplaceformula[e]
    \startformula
        (6)
    \stopformula
\stopplaceformula
```

$$(6) \tag{5.6}$$

In the next examples we demonstrate how we can avoid numbering, pass a reference as key, use assignments instead and add a title or suffix.

```
\startplaceformula
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[-]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[p]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[reference=foo]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[title=whatever]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[suffix=q]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[r]
    \startformula e=mc^2 \stopformula
\stopplaceformula
```

$$e = mc^2 \tag{5.7}$$

$$e = mc^2$$

$$e = mc^2 \tag{5.8}$$

$$e = mc^2 \tag{5.9}$$

$$e = mc^2 \qquad \text{(whatever)}$$

$$e = mc^2 \tag{5.10.q}$$

$$e = mc^2 \tag{5.11}$$

If you want consistent spacing you can enforce this:

```
\startplaceformula[s]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[-]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[-]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\setupformulas[numberstrut=always]
\startplaceformula[-]
    \startformula e=mc^2 \stopformula
\stopplaceformula
\startplaceformula[-]
    \startformula e=mc^2 \stopformula
\stopplaceformula
```

$$e = mc^2 \tag{5.12}$$

$$e = mc^2$$

$$e = mc^2$$

$$e = mc^2$$

$$e = mc^2$$

Possible values for `numberstrut` are `yes` (the default), `always` and `no`.

6 Combining formulas

Multiple formulas can be combined by wrapping them:

```
\fakewords{20}{30}

\startformula
    a + b = c
\stopformula

\fakewords{20}{30}

\startformulas
    \startformula
        a + b = c
    \stopformula
    \startformula
        d - e = f
    \stopformula
\stopformulas

\fakewords{20}{30}

\startformulas
    \startformula
        \frac{\frac{x}{y}}{b} = c
    \stopformula
    \startformula
        d - e = f
    \stopformula
\stopformulas

\fakewords{20}{30}
```

When we bump the space around formulas to big we get this:

[redacted]

$$a + b = c$$

[redacted]

$$a + b = c \qquad\qquad d - e = f$$

[redacted]

$$\frac{\frac{x}{y}}{b} = c \qquad d - e = f$$

[redacted]

The formulas get aligned on the baselline which in turn relates to the math axis of the formula.

7 Features

7.1 Default features

Math fonts are loaded in so called basemode, which gives them a traditional treatment in the engine. However, we do support features in basemode too, so setting them can influence what gets passed to TEX. Also, in math mode, some font features (like `dtls` and stylistic alternates) are applied dynamically.

The default `mathematics` feature set is as follows:

```
compactmath       yes
kern              yes
language          dflt
mathalternates    yes
mathdimensions    all
mathitalics       yes
mathnolimitsmode  0,800
mode              base
script            math
```

We don't discuss the exact meaning of these options here because normally you don't have to deal with them. If a math font demands something special, the place to deal with it is the related font goodie file.

This feature set is the parent of two other sets: `mathematics-l2r` and `mathematics-r2l`:

```
compactmath       yes
kern              yes
language          dflt
mathalternates    yes
mathdimensions    all
mathitalics       yes
mathnolimitsmode  0,800
mode              base
script            math
```

This one is the same as the parent but the right-to-left variant is different:

```
compactmath       yes
kern              yes
language          dflt
locl              yes
mathalternates    yes
mathdimensions    all
mathitalics       yes
mathnolimitsmode  0,800
```

mode	base
rtlm	yes
script	math

Eventually we need size related feature sets and again we define a parent and direction specific ones: `math-text`, `math-script` and `math-scriptscript`.

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mode	base
script	math
ssty	no

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mathsize	yes
mode	base
script	math
ssty	1

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mathsize	yes
mode	base
script	math
ssty	2

The left-to-right sets `math-*-l2r` are:

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mode	base
script	math
ssty	no

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mathsize	yes
mode	base
script	math
ssty	1

compactmath	yes
kern	yes
language	dflt
mathalternates	yes
mathdimensions	all
mathitalics	yes
mathnolimitsmode	0,800
mathsize	yes
mode	base
script	math
ssty	2

The right-to-left sets `math-*-r2l` are:

compactmath	yes
kern	yes
language	dflt
locl	yes
mathalternates	yes
mathdimensions	all
mathitalics	yes

```
mathnolimitsmode  0,800
mode              base
rtlm              yes
script            math
ssty              no

compactmath       yes
kern              yes
language          dflt
locl              yes
mathalternates    yes
mathdimensions    all
mathitalics       yes
mathnolimitsmode  0,800
mathsize          yes
mode              base
rtlm              yes
script            math
ssty              1

compactmath       yes
kern              yes
language          dflt
locl              yes
mathalternates    yes
mathdimensions    all
mathitalics       yes
mathnolimitsmode  0,800
mathsize          yes
mode              base
rtlm              yes
script            math
ssty              2
```

There are a few extra sets defined but these are meant for testing or virtual math fonts. The reason for showing these sets is to make clear that the number of features is minimal and that math is a real script indeed.

The `kern` features is questionable. In traditional TEX there are kerns indeed but in OpenType math kerns are not used that way because a more advanced kerning feature is present (and that one is currently always enabled). We used to set the following but these make no sense.

```
liga=yes, % (traditional) ligatures
tlig=yes, % tex ligatures, like -- and ---
```

```
trep=yes, % tex replacements, like the ' quote
```

Math fonts normally have no ligatures and supporting the TEX specific ones can actually be annoying. So, in todays ConTEXt these are no longer enabled. Just consider the following:

```
$- \kern0pt -   \kern 0pt \mathchar"2D$
$- \kern0pt --  \kern 0pt \mathchar"2D \mathchar"2D$
$- \kern0pt --- \kern 0pt \mathchar"2D \mathchar"2D \mathchar"2D$
```

The `-` is mapped onto a minus sign and therefore several in succession become multiple minus signs. The `\mathchar"2D` will remain the character with that slot in the font so there we will see a hyphen. If we would enable the `tlig` feature several such characters would be combined into an endash or emdash. So how do we get these than? Because getting a hyphen directly involves a command, the same is true for its longer relatives: `\endash` and `\emdash`.

− − -
− − −--
− − − − ---

As convenience we have defined a special `\mathhyphen` command. Watch the fact that a text hyphen in math mode is a minus in math! As comparison we also show the plus sign.

command	**math**	**text**
`\mathhyphen`	-	-
`\texthyphen`	−	-
`-`	−	-
`+`	+	+
`\endash`	–	–
`\emdash`	—	—

7.2 Stylistic alternates

todo

7.3 Dotless variants

todo

7.4 Script kerning

Text in math is somewhat special. First of all, a math font is not a text fonts because the characters and glyphs have a different purpose. Text features are normally not present (and often not even wanted). Anyway, you can force a text font, but that doesn't mean you will get for instance kerning. You can force a box which in turn will trigger font processing, but then you normally loose the script related size properties. So we end up with some juggling possibly combined with user intervention, and that is what the `\text` macro does.

But still there is the kern between a variable and its subscript to consider, something that normally is dealt with with staircase kerns, an OpenType math speciality. But, as we progress over the math list, and we bind a subscript to a variable, that subscript can be anything: a simple character, or more characters (a list) or something wrapped in a box. There is simply no universal solution that we can hard code because sometimes you don't want that special kerning. This is why in LuaTeX the integer variable `\mathscriptboxmode` controls the way this is dealt with.

- 0 forget about kerning
- 1 kern math sub lists with a valid glyph (default in the engine)
- 2 also kern math sub boxes that have a valid glyph (default in ConTeXt)
- 3 only kern math sub boxes with a boundary node present

Here we show some examples of how this parameter controls kerning. Watch the difference between a simple font switch and a text wrapped in a box. There are differences between fonts: some fonts have kerns, some don't. When present kerns are passed to the engine without further user intervention.

`$T_{\tf fluff}$` (mode 0)

modern	T_{fluff}
lucidaot	T_{fluff}
pagella	T_{fluff}
cambria	T_{fluff}
dejavu	T_{fluff}

`$T_{\tf fluff}$` (mode 1)

modern	T_{fluff}
lucidaot	T_{fluff}
pagella	T_{fluff}

cambria T_{fluff}

dejavu T_{fluff}

T_{fluff} (mode 1)

modern T_{fluff}

lucidaot T_{fluff}

pagella T_{fluff}

cambria T_{fluff}

dejavu T_{fluff}

T_{fluff} (mode 2)

modern T_{fluff}

lucidaot T_{fluff}

pagella T_{fluff}

cambria T_{fluff}

dejavu T_{fluff}

$T_{\text{\boundary1 fluff}}$ (mode 3)

modern T_{fluff}

lucidaot T_{fluff}

pagella T_{fluff}

cambria T_{fluff}

dejavu T_{fluff}

8 Alignments and such

8.1 Using ampersands

When you come from plain TEX, using ampersands probably comes as a custom, like in:

```
\startformula
\bordermatrix {
   a & b & c & d \cr
   e & f & G & h \cr
   i & j & k & l \cr
}
\stopformula
```

$$\begin{array}{l} a \mathbin{\&} b \mathbin{\&} c \mathbin{\&} d \\ e \mathbin{\&} f \mathbin{\&} G \mathbin{\&} h \\ i \mathbin{\&} j \mathbin{\&} k \mathbin{\&} l \end{array} \begin{pmatrix} \end{pmatrix}$$

or:

```
\startformula
\bbordermatrix {
   a & b & c & d \cr
   e & f & G & h \cr
   i & j & k & l \cr
}
\stopformula
```

$$\begin{array}{l} a \mathbin{\&} b \mathbin{\&} c \mathbin{\&} d \\ e \mathbin{\&} f \mathbin{\&} G \mathbin{\&} h \\ i \mathbin{\&} j \mathbin{\&} k \mathbin{\&} l \end{array} \begin{bmatrix} \end{bmatrix}$$

A more ConTEXt way is this:

```
\startformula
\startbordermatrix
   \NC a \NC b \NC c \NC d \NR
   \NC e \NC f \NC G \NC h \NR
   \NC i \NC j \NC k \NC l \NR
\stopbordermatrix
\stopformula
```

$$\begin{matrix} a & b & c & d \\ e & \multicolumn{3}{c}{\begin{pmatrix} f & G & h \\ j & k & l \end{pmatrix}} \\ i \end{matrix}$$

and:

```
\startformula
\startbbordermatrix
   \NC a \NC b \NC c \NC d \NR
   \NC e \NC f \NC G \NC h \NR
   \NC i \NC j \NC k \NC l \NR
\stopbbordermatrix
\stopformula
```

$$\begin{matrix} & a & b & c & d \\ e & \left[f \right. & G & \left. h \right] \\ i & \left[j \right. & k & \left. l \right] \end{matrix}$$

Just that you know. In general ampersands in ConTEXt text mode are just that: ampersands, not something alignment related.

8.2 Locations

The `location` feature gives some control over the alignment of alignments. The following examples are taken from an email exchange with Henri Menke.

```
\startplaceformula
  \startformula
    \startmathalignment[location=top]
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
    \stopmathalignment
    \quad\text{or}\quad
    \startmathalignment[location=center]
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
    \stopmathalignment
    \quad\text{or}\quad
    \startmathalignment[location=bottom]
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
    \stopmathalignment
  \stopformula
\stopplaceformula
```

$$\begin{array}{c} a+b=c+d \\ a+b=c+d \\ a+b=c+d \end{array} \quad \text{or} \quad \begin{array}{c} a+b=c+d \\ a+b=c+d \\ a+b=c+d \end{array} \quad \text{or} \quad \begin{array}{c} a+b=c+d \\ a+b=c+d \\ a+b=c+d \end{array} \qquad (8.1)$$

Numbering works ok for a single mathalignment

```
\startplaceformula
  \startformula
    \startmathalignment[number=auto]
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
    \stopmathalignment
  \stopformula
\stopplaceformula
```

$$a+b=c+d \qquad (8.2)$$

$$a+b=c+d \qquad (8.3)$$

$$a+b=c+d \qquad (8.4)$$

But for one with a location the results are suboptimal:

```
\startplaceformula
  \startformula
    \startmathalignment[location=center,number=auto]
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
      \NC a + b \NC= c + d \NR
    \stopmathalignment
  \stopformula
\stopplaceformula
```

$$a+b=c+d \, (8.5)$$

$$a+b=c+d \, (8.6)$$

$$a+b=c+d \, (8.7)$$

Here is a real example:

```
\startplaceformula
```

```
  \startformula
    U_2 = \frac{1}{2!}
      \int_0^\beta \diff\tau_1 \int_0^\beta \diff\tau_2\;
      \sum_{\startsubstack k_1,q_1 \NR k_2,q_2 \stopsubstack}
      \Bigl\langle
      \startmathalignment[location=top,align=left]
        \NC
          \mathcal T \Bigl[
            c_{k_1}^\dagger (\tau_1)
            \Delta_{k_1,q_1}^r c_{-k_1}^* (\tau_1) + c_{-q_1}^T (\tau_1)
            \Delta_{k_1,q_1}^{r\dagger} c_{q_1} (\tau_1)
          \Bigr]
        \NR
        \NC
          \times \Bigl[
            c_{k_2}^\dagger(\tau_2) \Delta_{k_2,q_2}^r c_{-k_2}^*
            (\tau_2) + c_{-q_2}^T (\tau_2) \Delta_{k_2,q_2}^{r\dagger}
            c_{q_2} (\tau_2)
          \Bigr] \Bigr\rangle .
        \NR
      \stopmathalignment
  \stopformula
\stopplaceformula
```

$$U_2 = \frac{1}{2!}\int_0^\beta d\tau_1 \int_0^\beta d\tau_2 \sum_{\substack{k_1,q_1\\k_2,q_2}} \Bigl\langle \begin{aligned}&\mathcal{T}\Bigl[c_{k_1}^\dagger(\tau_1)\Delta_{k_1,q_1}^r c_{-k_1}^*(\tau_1) + c_{-q_1}^T(\tau_1)\Delta_{k_1,q_1}^{r\dagger}c_{q_1}(\tau_1)\Bigr]\\&\times\Bigl[c_{k_2}^\dagger(\tau_2)\Delta_{k_2,q_2}^r c_{-k_2}^*(\tau_2) + c_{-q_2}^T(\tau_2)\Delta_{k_2,q_2}^{r\dagger}c_{q_2}(\tau_2)\Bigr]\Bigr\rangle.\end{aligned} \tag{8.8}$$

8.3 Tuning alignments

Again a few examples of manipulating alignments. It really helps to play with examples if you want to get an idea what is possible.

```
\startformula
    \startalign[m=2,align={middle}]
        \NC \text to 6cm{} \NC x = 0 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={middle}]
```

```
        \NC \text to 6cm{One\hfill}           \NC a = 1 \NR
        \NC \text to 6cm{One Two\hfill}       \NC b = 2 \NR
        \NC \text to 6cm{One Two Three\hfill} \NC c = 3 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={left}]
        \NC \text to 6cm{One\hfill}           \NC a = 1 \NR
        \NC \text to 6cm{One Two\hfill}       \NC b = 2 \NR
        \NC \text to 6cm{One Two Three\hfill} \NC c = 3 \NR
    \stopalign
\stopformula
```

$$x = 0$$

One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$
One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$

```
\startformula
    \startalign[m=2,align={middle}]
        \NC \text to 6cm{} \NC x = 0 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={middle}]
        \NC \text to 6cm{One}           \NC a = 1 \NR
        \NC \text to 6cm{One Two}       \NC b = 2 \NR
        \NC \text to 6cm{One Two Three} \NC c = 3 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={left}]
        \NC \text to 6cm{One}           \NC a = 1 \NR
        \NC \text to 6cm{One Two}       \NC b = 2 \NR
```

```
        \NC \text to 6cm{One Two Three} \NC c = 3 \NR
    \stopalign
\stopformula
```

$$x = 0$$

One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$

One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$

```
\startformula
    \startalign[m=2,align={middle}]
        \NC \text{} \NC x = 0 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={middle}]
        \NC \text{One}           \NC a = 1 \NR
        \NC \text{One Two}       \NC b = 2 \NR
        \NC \text{One Two Three} \NC c = 3 \NR
    \stopalign
\stopformula

\startformula
    \startalign[m=2,align={left}]
        \NC \text{One}           \NC a = 1 \NR
        \NC \text{One Two}       \NC b = 2 \NR
        \NC \text{One Two Three} \NC c = 3 \NR
    \stopalign
\stopformula
```

$$x = 0$$

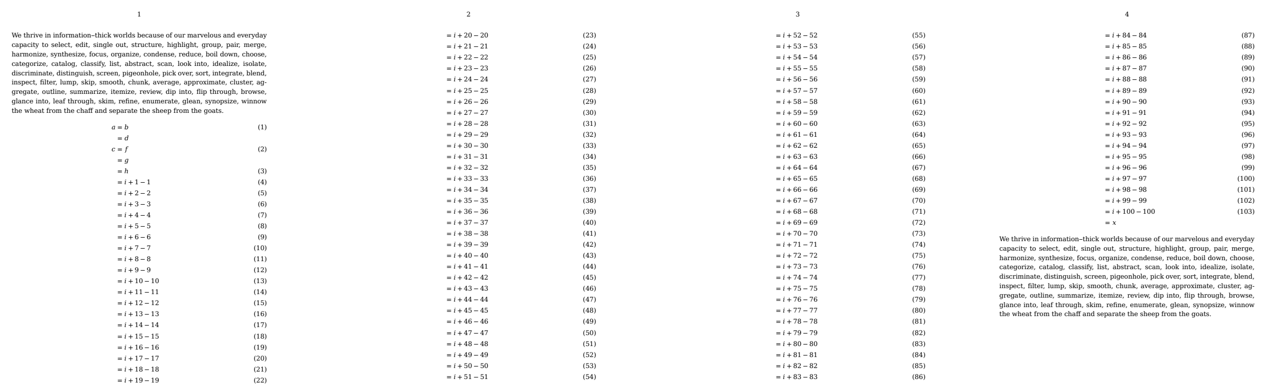
One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$

One	$a = 1$
One Two	$b = 2$
One Two Three	$c = 3$

8.4 Splitting over pages

Because formula placement has positioning options a formula gets wrapped in a box. As a consequence formulas will not break across pages. This can be an issue with alignments. There is an experimental option for this (the result is shown in figure 8.1):

```
\usemodule[art-01]
\setupbodyfont[13pt]
\starttext
  \input tufte
  \startplaceformula
    \startsplitformula
      \startalign
        \NC a \EQ b \NR[+]
        \NC   \EQ d \NR
        \NC c \EQ f \NR[+]
        \NC   \EQ g \NR
        \NC   \EQ h \NR[+]
        \dorecurse{100}{\NC \EQ i + #1 - #1\NR[+]}%
        \NC   \EQ x \NR
      \stopalign
    \stopsplitformula
  \stopplaceformula
  \input tufte
\stoptext
```



1

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

$$\begin{aligned}
a &= b \tag{1}\\
&= d \\
c &= f \tag{2}\\
&= g \\
&= h \tag{3}\\
&= i + 1 - 1 \tag{4}\\
&= i + 2 - 2 \tag{5}\\
&= i + 3 - 3 \tag{6}\\
&= i + 4 - 4 \tag{7}\\
&= i + 5 - 5 \tag{8}\\
&= i + 6 - 6 \tag{9}\\
&= i + 7 - 7 \tag{10}\\
&= i + 8 - 8 \tag{11}\\
&= i + 9 - 9 \tag{12}\\
&= i + 10 - 10 \tag{13}\\
&= i + 11 - 11 \tag{14}\\
&= i + 12 - 12 \tag{15}\\
&= i + 13 - 13 \tag{16}\\
&= i + 14 - 14 \tag{17}\\
&= i + 15 - 15 \tag{18}\\
&= i + 16 - 16 \tag{19}\\
&= i + 17 - 17 \tag{20}\\
&= i + 18 - 18 \tag{21}\\
&= i + 19 - 19 \tag{22}
\end{aligned}$$

2

$$\begin{aligned}
&= i + 20 - 20 \tag{23}\\
&= i + 21 - 21 \tag{24}\\
&= i + 22 - 22 \tag{25}\\
&= i + 23 - 23 \tag{26}\\
&= i + 24 - 24 \tag{27}\\
&= i + 25 - 25 \tag{28}\\
&= i + 26 - 26 \tag{29}\\
&= i + 27 - 27 \tag{30}\\
&= i + 28 - 28 \tag{31}\\
&= i + 29 - 29 \tag{32}\\
&= i + 30 - 30 \tag{33}\\
&= i + 31 - 31 \tag{34}\\
&= i + 32 - 32 \tag{35}\\
&= i + 33 - 33 \tag{36}\\
&= i + 34 - 34 \tag{37}\\
&= i + 35 - 35 \tag{38}\\
&= i + 36 - 36 \tag{39}\\
&= i + 37 - 37 \tag{40}\\
&= i + 38 - 38 \tag{41}\\
&= i + 39 - 39 \tag{42}\\
&= i + 40 - 40 \tag{43}\\
&= i + 41 - 41 \tag{44}\\
&= i + 42 - 42 \tag{45}\\
&= i + 43 - 43 \tag{46}\\
&= i + 44 - 44 \tag{47}\\
&= i + 45 - 45 \tag{48}\\
&= i + 46 - 46 \tag{49}\\
&= i + 47 - 47 \tag{50}\\
&= i + 48 - 48 \tag{51}\\
&= i + 49 - 49 \tag{52}\\
&= i + 50 - 50 \tag{53}\\
&= i + 51 - 51 \tag{54}
\end{aligned}$$

3

$$\begin{aligned}
&= i + 52 - 52 \tag{55}\\
&= i + 53 - 53 \tag{56}\\
&= i + 54 - 54 \tag{57}\\
&= i + 55 - 55 \tag{58}\\
&= i + 56 - 56 \tag{59}\\
&= i + 57 - 57 \tag{60}\\
&= i + 58 - 58 \tag{61}\\
&= i + 59 - 59 \tag{62}\\
&= i + 60 - 60 \tag{63}\\
&= i + 61 - 61 \tag{64}\\
&= i + 62 - 62 \tag{65}\\
&= i + 63 - 63 \tag{66}\\
&= i + 64 - 64 \tag{67}\\
&= i + 65 - 65 \tag{68}\\
&= i + 66 - 66 \tag{69}\\
&= i + 67 - 67 \tag{70}\\
&= i + 68 - 68 \tag{71}\\
&= i + 69 - 69 \tag{72}\\
&= i + 70 - 70 \tag{73}\\
&= i + 71 - 71 \tag{74}\\
&= i + 72 - 72 \tag{75}\\
&= i + 73 - 73 \tag{76}\\
&= i + 74 - 74 \tag{77}\\
&= i + 75 - 75 \tag{78}\\
&= i + 76 - 76 \tag{79}\\
&= i + 77 - 77 \tag{80}\\
&= i + 78 - 78 \tag{81}\\
&= i + 79 - 79 \tag{82}\\
&= i + 80 - 80 \tag{83}\\
&= i + 81 - 81 \tag{84}\\
&= i + 82 - 82 \tag{85}\\
&= i + 83 - 83 \tag{86}
\end{aligned}$$

4

$$\begin{aligned}
&= i + 84 - 84 \tag{87}\\
&= i + 85 - 85 \tag{88}\\
&= i + 86 - 86 \tag{89}\\
&= i + 87 - 87 \tag{90}\\
&= i + 88 - 88 \tag{91}\\
&= i + 89 - 89 \tag{92}\\
&= i + 90 - 90 \tag{93}\\
&= i + 91 - 91 \tag{94}\\
&= i + 92 - 92 \tag{95}\\
&= i + 93 - 93 \tag{96}\\
&= i + 94 - 94 \tag{97}\\
&= i + 95 - 95 \tag{98}\\
&= i + 96 - 96 \tag{99}\\
&= i + 97 - 97 \tag{100}\\
&= i + 98 - 98 \tag{101}\\
&= i + 99 - 99 \tag{102}\\
&= i + 100 - 100 \tag{103}\\
&= x
\end{aligned}$$

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

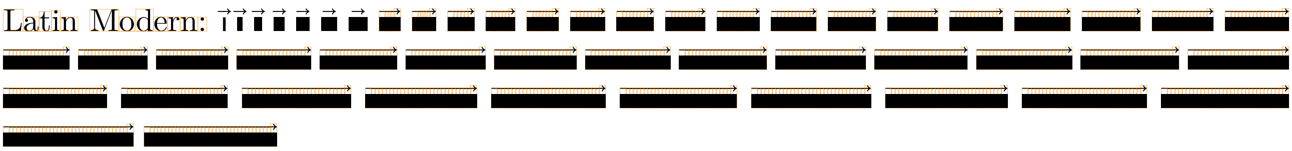
Figure 8.1 Splitting an alignment.

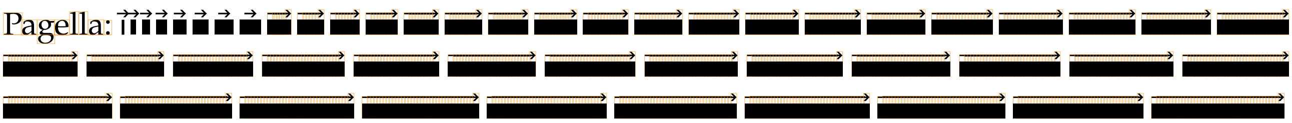
9 Suboptimal

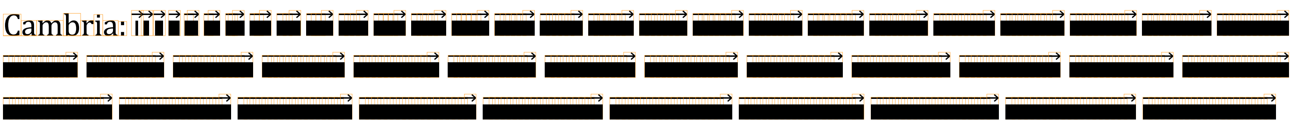
9.1 Extensibles

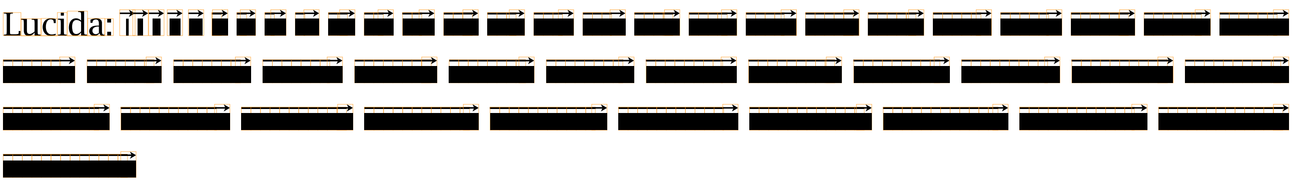
Extensibles are implemented as follows: we start with the default shape, and when that doesn't cover the body of text, a next size is chosen. When we run out of sizes, a glyph is made from snippets (often a start glyph, overlapping middle pieces and an end piece. Of course a font needs to provide these variants and snippets.

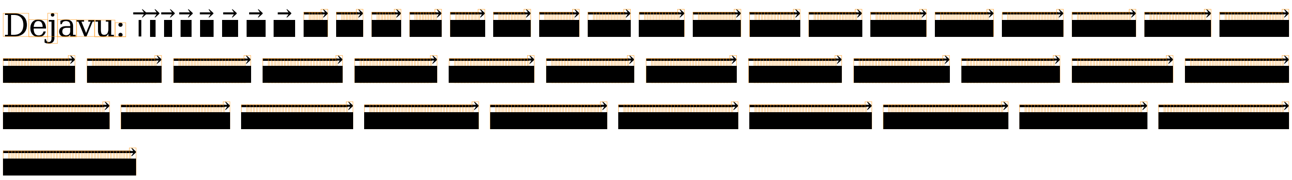
However, the quality of the coverage can differ per font. Here we show how Latin Modern, Pagella, Cambria, Lucida and Dejavu look like:

Latin Modern:


Pagella:


Cambria:


Lucida:


Dejavu:


Of course fonts can be improved (or patched) and these samples might come out better compared to previous renderings.

10 Tricks

10.1 Introduction

Math support in ConTEXt is wrapped around basic TEX primitives and unfortunately not all we want is easy to configure. This is not surprising because the original ideas behind TEX are that one makes a style per book and a one macro package 'we-can-do-it-all' approach is not what Don Knuth had in mind at that time.

So, for instance support for configurable spacing per math element, coloring of specific (sub) elements, simple switching of whatever combination of alignments and number placement, these all take quite a bit of code and hackery.

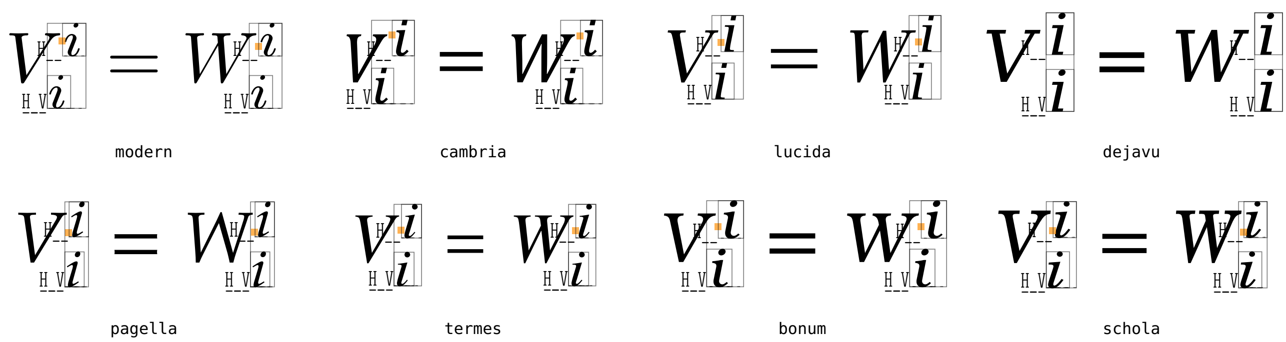
Even configuring something seemingly trivial as fractions or top, bottom, left, middle and right fences take some effort. This is because the engine uses information from fonts to combine shapes and paste the content and ornaments to together.

For that reason already in MkII but more extensively in MkIV we did a lot of these things in wrapper macros. When the math renderer was finalized for OpenType math some extra control was added that can make these things easier. However, because we go a bit beyond what is possible using this new functionality these new mechanisms are not yet used in MkIV, but they might be eventually. Here we just show some of the (newer) low level trickery. For details about what was already possible in pure TEX, we refer to the ultimate references: the TEXbook (by Donald Knuth) and TEX by Topic (by Victor Eijkhout).

10.2 Kerning

Kerning in OpenType math is not the same as in traditional TEX: instead of a single value, we have staircase kerns, that is, depending on the location (left or right) and the vertical position, at discrete distances between depth and height. In addition there is italic correction but that is only applied in certain cases, one of which is the script location.

Unfortunately not all fonts follow the same route. Some fonts have a true width and a moderate italic correction is added to it (of at all), while other fonts lie about the width and depend on an excessive italic correction to compensate for that.



I will not discuss the details because when a font gets updated, it might look better or worse. These fonts were loaded with the following directive set:

```
\enabledirectives[fontgoodies.mathkerning]
```

An example of a fontgoodie that fixed the kerning is `pagella-math.lfg`. Here is the relevant bit:

```
local kern_200 = { bottomright = { { kern = -200 } } }
local kern_100 = { bottomright = { { kern = -100 } } }

return {
  .....
  mathematics = {
    .....
    kerns = {
      [0x1D449] = kern_200, --
      [0x1D44A] = kern_100, --
    },
    .....
  }
}
```

This fixes the real bad kerning of Pagella Math which at least in 2017 was not (yet) fixed. When the fonts are frozen we can start makling permanent runtime fixes like this.

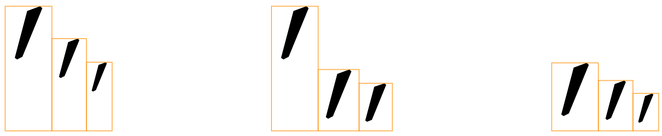
10.3 Primes

Primes are a pain in the butt. The reason for this is that they are independent characters on the one hand but can be seen as a superscript on the other. Let's first look at the symbols at the three sizes that are used in math.

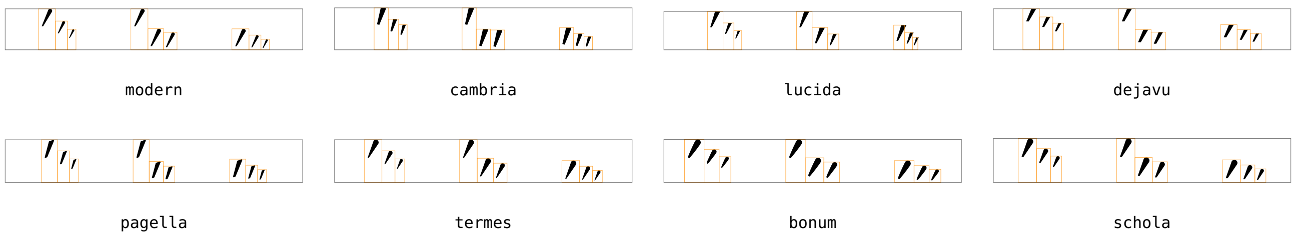
```
$
    {\textstyle        \char"2032}
    {\scriptstyle      \char"2032}
    {\scriptscriptstyle\char"2032}
\quad
    {\textstyle        \char"FE931}
    {\scriptstyle      \char"FE931}
    {\scriptscriptstyle\char"FE931}
\quad
    {\textstyle        \char"FE932}
    {\scriptstyle      \char"FE932}
```

```
    {\scriptscriptstyle\char"FE932}
$
```

We blow up the characters a bit and get this:



The first set is the normal prime character scaled to the text, script and scriptscriptsize. The second set shows the characters (at three sizes) as they are in the font. The largest character is raised while the other two are closer to the baseline. In some fonts the smaller sizes arenot smaller at all. The last set is a variant of the the first set but we made them into virtual characters with a displacement and different dimensions. Those are the ones we use as primes.



Next we show how primes show up in real math. The examples explain themselves.

```
{\textstyle         f = g} \quad
{\scriptstyle       f = g} \quad
{\scriptscriptstyle f = g}
```

$$f = g \quad {\scriptstyle f = g} \quad {\scriptscriptstyle f = g}$$

```
{\textstyle         f_i' = g_i'} \quad
{\scriptstyle       f_i' = g_i'} \quad
{\scriptscriptstyle f_i' = g_i'}
```

$$f_i' = g_i' \quad {\scriptstyle f_i' = g_i'} \quad {\scriptscriptstyle f_i' = g_i'}$$

```
{\textstyle         f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptstyle       f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptscriptstyle f^{\char"2032}(0) = g^{\char"2032}(0)}
```

$$f^{\prime}(0) = g^{\prime}(0) \quad {\scriptstyle f^{\prime}(0) = g^{\prime}(0)} \quad {\scriptscriptstyle f^{\prime}(0) = g^{\prime}(0)}$$

```
{\textstyle         f'(0) = g'(0)} \quad
{\scriptstyle       f'(0) = g'(0)} \quad
{\scriptscriptstyle f'(0) = g'(0)}
```

$$\textstyle f'(0) = g'(0) \quad {\scriptstyle f'(0) = g'(0)} \quad {\scriptscriptstyle f'(0) = g'(0)}$$

```
{\textstyle         f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptstyle       f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptscriptstyle f^{\char"2032}(0) = g^{\char"2032}(0)}
```

$$\textstyle f^{\prime}(0) = g^{\prime}(0) \quad {\scriptstyle f^{\prime}(0) = g^{\prime}(0)} \quad {\scriptscriptstyle f^{\prime}(0) = g^{\prime}(0)}$$

```
{\textstyle         f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptstyle       f^{\char"2032}(0) = g^{\char"2032}(0)} \quad
{\scriptscriptstyle f^{\char"2032}(0) = g^{\char"2032}(0)}
```

$$\textstyle f^{\prime}(0) = g^{\prime}(0) \quad {\scriptstyle f^{\prime}(0) = g^{\prime}(0)} \quad {\scriptscriptstyle f^{\prime}(0) = g^{\prime}(0)}$$

The prime analyzer can deal with sizes, subscripts but also converts a sequence of upright quotes into one unicode symbol. So,

```
f'_i \neq f''_i \neq f'''_i \neq f''''_i
```

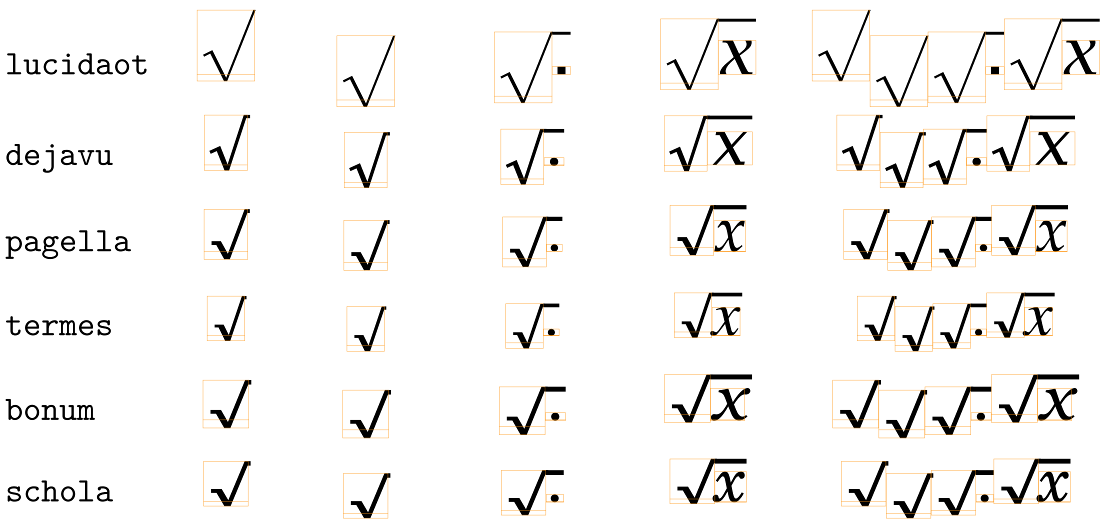
becomes:

$$f'_i \neq f''_i \neq f'''_i \neq f''''_i$$

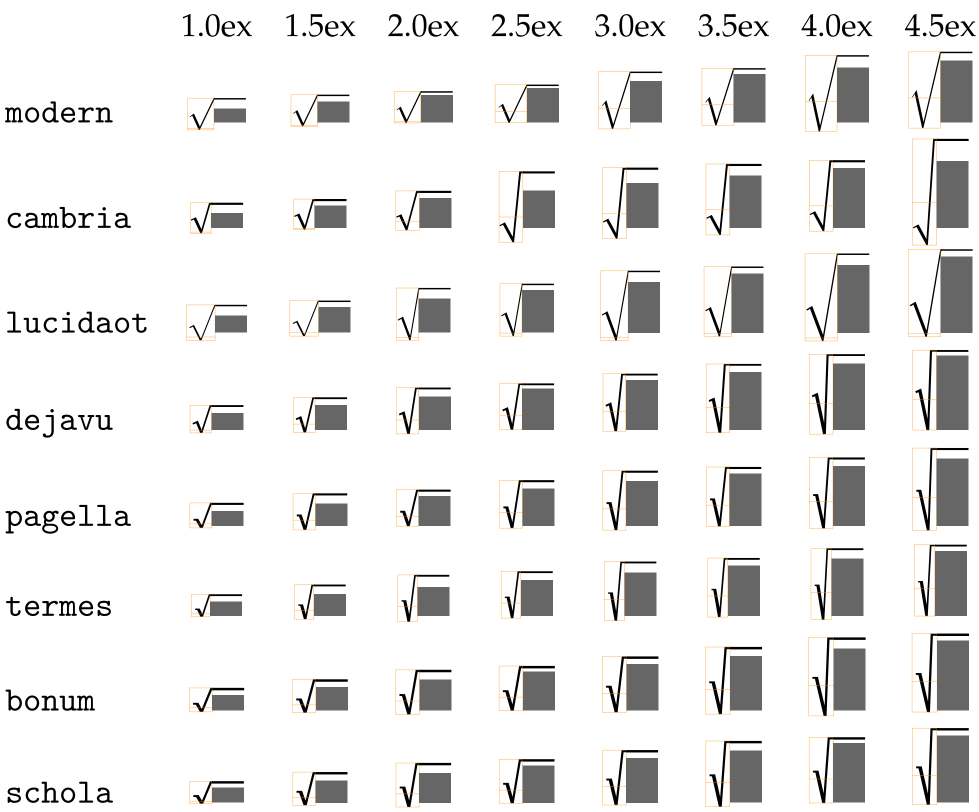
10.4 Radicals

Sometimes users complain about the look of a radical symbol. This is however a matter of design. Some fonts let the shape start more below the baseline than others. Soem go more straight up than relatives in another font. When largers sizes are needed, some fonts offer smaller than others. Just look at the different desings:

	\surd	\sqrt{}	\sqrt{.}	\sqrt{x}	
modern	$\surd$	$\sqrt{}$	$\sqrt{.}$	$\sqrt{x}$	$\surd\sqrt{}\sqrt{.}\sqrt{x}$
cambria	$\surd$	$\sqrt{}$	$\sqrt{.}$	$\sqrt{x}$	$\surd\sqrt{}\sqrt{.}\sqrt{x}$



The automatic scaling doesn't always work out as expected but on the average is okay. Keep in mind that often the content is not that extreme.



In Lucida (the version at the time of writing this) we have to correct the threshold a bit in the goodie file:

```
local function FixRadicalDisplayStyleVerticalGap(value,target,original)
  local o = original.mathparameters.RadicalVerticalGap -- 50
  return 2 * o * target.parameters.factor
end
```

```
return {
  .....
  mathematics = {
    .....
    parameters = {
      RadicalDisplayStyleVerticalGap =
        FixRadicalDisplayStyleVerticalGap,
    },
    .....
  },
}
```

10.5 Integrals

This section needs to be adapted to extensible integrals.

A curious exception in the math system is the integral sign. Its companions are the summation and product signs, but integral has as extra property that it has a slant. In LuaTeX there is rather advanced control over how the (optional) scripts are positioned (which relates to italic correction) but in ConTeXt we only make limited use of that. The main reason is that we also need to support additional features like color. Therefore integrals are handled by the extensible mechanism.

The size of an integral is more of less fixed but you can enlarge to your liking. One reason for this is that you might want a consistent size across formulas. Let's use the following setup:

```
% \setupmathextensible
\setupmathdelimiter
  [integral]
  [rightoffset=-1mu,
   exact=yes,
   factor=2]
```

We use the following exmaple:

```
\ruledhbox{$\integral                   f\frac{1}{2}             $}\quad
\ruledhbox{$\integral[rightoffset=3mu] f\frac{1}{2}             $}\quad
\ruledhbox{$\integral[exact=no]        f\frac{1}{2}             $}\quad
\ruledhbox{$\integral                   f\frac{\frac{1}{2}}{x} $}\quad
\ruledhbox{$\integral[exact=no]        f\frac{\frac{1}{2}}{x} $}\quad
\ruledhbox{$\integral[factor=1]        f\frac{1}{2}             $}\quad
\ruledhbox{$\integral[factor=3]        f\frac{\frac{1}{2}}{x} $}\quad
\ruledhbox{$\integral[factor=3]        f\frac{1}{2}             $}\quad
```

```
\ruledhbox{$\int                                    f\frac{1}{2}             $}% bonus
```

This renders as:

$\int f \frac{1}{2}$ $\int [rightoffset = 3mu] f \frac{1}{2}$ $\int [exact = no] f \frac{1}{2}$ $\int f \frac{\frac{1}{2}}{x}$ $\int [exact = no] f \frac{\frac{1}{2}}{x}$ $\int [factor = 1] f \frac{1}{2}$ $\int [f a$

10.6 Fancy fences

Here I only show an example of fences drawn by MetaPost. For the implementation you can consult the library file `meta-imp-mat.mkiv` in the ConTeXt distribution.

```
\useMPlibrary[mat]

\setupmathstackers
  [both] % vfenced]
  [color=darkred,
   alternative=mp]

\setupmathstackers
  [top]
  [color=darkred,
   alternative=mp]

\setupmathstackers
  [bottom]
  [color=darkred,
   alternative=mp]
```

We keep the demo simple:

```
$ \overbracket     {a+b+c+d} \quad
  \underbracket    {a+b+c+d} \quad
  \doublebracket   {a+b+c+d} \quad
  \overparent      {a+b+c+d} \quad
  \underparent     {a+b+c+d} \quad
  \doubleparent    {a+b+c+d} $ \blank
$ \overbrace       {a+b+c+d} \quad
  \underbrace      {a+b+c+d} \quad
  \doublebrace     {a+b+c+d} \quad
  \overbar         {a+b+c+d} \quad
  \underbar        {a+b+c+d} \quad
  \doublebar       {a+b+c+d} $ \blank
$ \overleftarrow   {a+b+c+d} \quad
```

```
\overrightarrow  {a+b+c+d} \quad
\underleftarrow  {a+b+c+d} \quad
\underrightarrow {a+b+c+d} $ \blank
```

Or visualized:

$\overline{a+b+c+d}$ $\underline{a+b+c+d}$ $\overline{\underline{a+b+c+d}}$ $\overline{a+b+c+d}$ $\underline{a+b+c+d}$ $\overline{\underline{a+b+c+d}}$

$\overbrace{a+b+c+d}$ $\underbrace{a+b+c+d}$ $\overbrace{\underbrace{a+b+c+d}}$ $\overline{a+b+c+d}$ $\underline{a+b+c+d}$ $\overline{\underline{a+b+c+d}}$

$\overleftarrow{a+b+c+d}$ $\overrightarrow{a+b+c+d}$ $\underleftarrow{a+b+c+d}$ $\underrightarrow{a+b+c+d}$

10.7 Combined characters

We have some magic built with respect to sequences of characters. They are derived from information in the character database that ships with ConTEXt and are implemented as a sort of ligatures. Some are defined in Unicode, others are defined explicitly.

ml	U+02016	‖	U+0007C U+0007C	--	‖	\Vert \Arrowvert \lVert \rVert \doubleverticalbar
sp	U+02026	…	U+0002E U+0002E U+0002E	...	…	\ldots \dots
sp	U+02033	″	U+02032 U+02032		″	\doubleprime
sp	U+02034	‴	U+02032 U+02032 U+02032		‴	\tripleprime
sp	U+02036	‶	U+02035 U+02035		‶	\reverseddoubleprime
sp	U+02037	‷	U+02035 U+02035 U+02035		‷	\reversedtripleprime
sp	U+02057	⁗	U+02032 U+02032 U+02032 U+02032		⁗	\quadrupleprime
ml	U+02190	←	U+0003C U+02212	<-	< −	\leftarrow \gets \underleftarrow \overleftarrow
ml	U+02192	→	U+02212 U+0003E	->	− >	\rightarrow \to \underrightarrow \overrightarrow
ml	U+02194	↔	U+0003C U+02212 U+0003E	<->	< − >	\leftrightarrow
sp	U+0219A	↚	U+02190 U+00338	←	↚	\nleftarrow
sp	U+0219B	↛	U+02192 U+00338	→	↛	\nrightarrow
sp	U+021AE	↮	U+02194 U+00338		↮	\nleftrightarrow
sp	U+021CD	⇍	U+021D0 U+00338		⇍	\nLeftarrow
sp	U+021CE	⇎	U+021D4 U+00338		⇎	\nLeftrightarrow
sp	U+021CF	⇏	U+021D2 U+00338		⇏	\nRightarrow
ml	U+021D0	⇐	U+0003C U+0003D U+0003D	<==	<= =	\Leftarrow

ml	U+021D2	⇒	U+0003D U+0003D U+0003E	==>	= ≽	\Rightarrow
						\imply
ml	U+021D4	⇔	U+0003C U+0003D U+0003D U+0003E	<==>	<= ≽	\Leftrightarrow
sp	U+02204	∄	U+02203 U+00338		∄	\nexists
sp	U+02209	∉	U+02208 U+00338		∉	\notin
						\nin
sp	U+0220C	∌	U+0220B U+00338		∌	\nni
						\nowns
sp	U+02224	∤	U+02223 U+00338		∤	\ndivides
						\nmid
sp	U+02226	∦	U+02225 U+00338		∦	\nparallel
sp	U+0222C	∬	U+0222B U+0222B		∬	\iint
						\iintop
sp	U+0222D	∭	U+0222B U+0222B U+0222B		∭	\iiint
						\iiintop
sp	U+0222F	∯	U+0222E U+0222E		∯	\oiint
sp	U+02230	∰	U+0222E U+0222E U+0222E		∰	\oiiint
ml	U+02237	∷	U+0003A U+0003A	::	∷	\squaredots
ml	U+02239	∹	U+02212 U+0003A	−:	− :	\minuscolon
sp	U+02241	≁	U+0223C U+00338		≁	\nsim
sp	U+02244	≄	U+02243 U+00338		≄	\nsimeq
sp	U+02247	≇	U+02245 U+00338		≇	\approxnEq
sp	U+02249	≉	U+02248 U+00338		≉	\napprox
ml	U+02254	≔	U+0003A U+0003D	:=	:=	\colonequals
ml	U+02255	≕	U+0003D U+0003A	=:	=:	\equalscolon
sp	U+02260	≠	U+0003D U+00338	=	≠	\neq \ne
ml	U+02260	≠	U+0002F U+0003D	/=	/=	\neq \ne
ml	U+02261	≡	U+0003D U+0003D	==	= =	\equiv
sp	U+02262	≢	U+02261 U+00338		≢	\nequiv
ml	U+02262	≢	U+0002F U+0003D U+0003D	/==	/= =	\nequiv
ml	U+02264	≤	U+0003C U+0003D	<=	<=	\leq \le
ml	U+02265	≥	U+0003E U+0003D	>=	>=	\geq \ge
ml	U+0226A	≪	U+0003C U+0003C	<<	<<	\ll
ml	U+0226B	≫	U+0003E U+0003E	>>	>>	\gg
sp	U+0226D	≭	U+0224D U+00338		≭	\nasymp
ml	U+0226D	≭	U+0002F U+0224D	/	/≍	\nasymp
sp	U+0226E	≮	U+0003C U+00338	<	≮	\nless
ml	U+0226E	≮	U+0002F U+0003C	/<	/<	\nless
sp	U+0226F	≯	U+0003E U+00338	>	≯	\ngtr
ml	U+0226F	≯	U+0002F U+0003E	/>	/>	\ngtr
sp	U+02270	≰	U+02264 U+00338		≰	\nleq
ml	U+02270	≰	U+0002F U+0003C U+0003D	/<=	/<=	\nleq
sp	U+02271	≱	U+02265 U+00338		≱	\ngeq
ml	U+02271	≱	U+0002F U+0003E U+0003D	/>=	/>=	\ngeq
sp	U+02274	≴	U+02272 U+00338		≴	\nlesssim
sp	U+02275	≵	U+02273 U+00338		≵	\ngtrsim
sp	U+02278	≸	U+02276 U+00338		≸	\nlessgtr
sp	U+02279	≹	U+02277 U+00338		≹	\ngtrless
sp	U+02280	⊀	U+0227A U+00338		⊀	\nprec
sp	U+02281	⊁	U+0227B U+00338		⊁	\nsucc
sp	U+02284	⊄	U+02282 U+00338		⊄	\nsubset

sp	U+02285	⊅	U+02283 U+00338		⊅	\nsupset
sp	U+02288	⊈	U+02286 U+00338		⊈	\nsubseteq
sp	U+02289	⊉	U+02287 U+00338		⊉	\nsupseteq
sp	U+022AC	⊬	U+022A2 U+00338		⊬	\nvdash
sp	U+022AD	⊭	U+022A8 U+00338		⊭	\nvDash
sp	U+022AE	⊮	U+022A9 U+00338		⊮	\nVdash
sp	U+022AF	⊯	U+022AB U+00338		⊯	\nVDash
ml	U+022D8	⋘	U+0003C U+0003C U+0003C	<<<	<<<	\lll
						\llless
ml	U+022D9	⋙	U+0003E U+0003E U+0003E	>>>	>>>	\ggg
						\gggtr
ml	U+022DC	⋜	U+0003D U+0003C	=<	=<	\eqless
ml	U+022DD	⋝	U+0003D U+0003E	=>	=>	\eqgtr
sp	U+022E0	⋠	U+0227C U+00338		⋠	\npreccurlyeq
sp	U+022E1	⋡	U+0227D U+00338		⋡	\nsucccurlyeq
sp	U+022E2	⋢	U+02291 U+00338		⋢	\nsqsubseteq
sp	U+022E3	⋣	U+02292 U+00338		⋣	\nsqsupseteq
sp	U+022EA	⋪	U+022B2 U+00338		⋪	\ntriangleright
sp	U+022EB	⋫	U+022B3 U+00338		⋫	\ntriangleleft
sp	U+022EC	⋬	U+022B4 U+00338		⋬	\ntrianglelefteq
sp	U+022ED	⋭	U+022B5 U+00338		⋭	\ntrianglerighteq
ml	U+027F5	⟵	U+0003C U+02212 U+02212	<--	< −−	\longleftarrow
ml	U+027F6	⟶	U+02212 U+02212 U+0003E	-->	−− >	\longrightarrow
ml	U+027F7	⟷	U+0003C U+02212 U+02212 U+0003E	<-->	< −− >	\longleftrightarrow
ml	U+027F8	⟸	U+0003C U+0003D U+0003D U+0003D	<===	<= =	\Longleftarrow
ml	U+027F9	⟹	U+0003D U+0003D U+0003D U+0003E	===>	= ≽	\Longrightarrow
ml	U+027FA	⟺	U+0003C U+0003D U+0003D U+0003D U+0003E	<===>	<= ≽	\Longleftrightarrow
ml	U+02980	⦀	U+0007C U+0007C U+0007C	|	|||	\tripleverticalbar
sp	U+02A0C	⨌	U+0222B U+0222B U+0222B U+0222B		⨌	\iiiint
						\iiiintop
sp	U+02A74	?	U+0003A U+0003A U+0003D	::=		\coloncolonequals
sp	U+02A75	= =	U+0003D U+0003D	==	= =	\eqeq
sp	U+02A76	= =	U+0003D U+0003D U+0003D	===	= =	\eqeqeq
ml	U+02A8B	⪋	U+0003C U+0003D U+0003E	<=>	<=>	\lesseqqgtr
ml	U+02A8C	⪌	U+0003E U+0003D U+0003C	>=<	>=<	\gtreqqless

10.8 Middle class fences

The next examples are somewhat obscure. They are a side effect of some extensions to the engine that were introduced to control spacing around the `\middle` class fences. Actually there is no real middle class and spacing was somewhat hard codes when `\middle` was added to ε-TEX. In LuaTEX we have introduced keywords to some primitives that control spacing and other properties. This permits better control over spacing than messing around with (for instance) injected `\mathrel` commands that can have their own side effects.

We use the following definitions:

```
\def\Middle{\middle|}
```

```
\def\Riddle{\Umiddle class 5 |}
\def\Left  {\left  (}
\def\Right {\right )}
\def\Rel   {\mathrel{}}
\def\Per   {\mathrel{.}}
```

Applied to samples these give the following outcome and spacing:

`$ a b $`	*ab*
`$ \Rel a\Rel b\Rel $`	*a* *b*
`$ a b $`	*ab*
`$ \Per a\Per b\Per $`	. *a* . *b* .
`$\Left a \Middle b \Right$`	(*a*|*b*)
`$\Left\Rel a \Middle\Rel b\Rel\Right$`	(*a*| *b*)
`$\Left a \Middle b \Right$`	(*a*|*b*)
`$\Left\Rel a \Middle\Per b\Per\Right$`	(*a*| . *b* .)
`$\Left a \Middle b \Right$`	(*a*|*b*)
`$\Left\Rel a\Rel\Middle\Rel b\Rel\Right$`	(*a* | *b*)
`$\Left a \Middle b \Right$`	(*a*|*b*)
`$\Left\Per a\Per\Middle\Per b\Per\Right$`	(. *a* .| . *b* .)
`$\Left a \Riddle b \Right$`	(*a*|*b*)
`$\Left\Rel a \Riddle\Rel b\Rel\Right$`	(*a*| *b*)
`$\Left a \Riddle b \Right$`	(*a*|*b*)
`$\Left\Rel a \Riddle\Per b\Per\Right$`	(*a*| . *b* .)
`$\Left a \Riddle b \Right$`	(*a*|*b*)
`$\Left\Rel a\Rel\Riddle\Rel b\Rel\Right$`	(*a* | *b*)
`$\Left a \Riddle b \Right$`	(*a*|*b*)
`$\Left\Per a\Per\Riddle\Per b\Per\Right$`	(. *a* .| . *b* .)

10.9 Auto-punctuation

The `\setupmathematics` command has an option `autopunctuation` that influences the way spacing after punctuatuon is handled, especially in cases like the following (coordinates and such):

`no`	`yes`	`yes,semicolon`	`all`	`all,semicolon`
$(1, 2) = (1, 2)$	$(1,2) = (1,2)$	$(1,2) = (1,2)$	$(1,2) = (1,2)$	$(1,2) = (1,2)$
$(1.2) = (1.2)$	$(1.2) = (1.2)$	$(1.2) = (1.2)$	$(1.2) = (1.2)$	$(1.2) = (1.2)$
$(1; 2) = (1; 2)$	$(1; 2) = (1; 2)$	$(1;2) = (1;2)$	$(1; 2) = (1; 2)$	$(1;2) = (1;2)$

11 Things you might forget

11.1 Ampersands

You can skip this, but if you continue reading, here is some low level plain code (don't use this in ConTEXt):

```
\def\matrix#1%
  {\null
   \,
   \vcenter
     {\normalbaselines
      \ialign{\hfil$##$\hfil && \quad\hfil$##$\hfil\crcr
      \mathstrut\crcr
      \noalign{\kern-\baselineskip}
      #1\crcr
      \mathstrut\crcr
      \noalign{\kern-\baselineskip}}}%
   \,}
```

You see the & here and it's the alignment cell separator. The special meaning of these characters is determined by the so called catcode. Here we have:

```
\catcode"26=4
```

Character `0x26` is the ampersand. In ConTEXt this character can be used in text mode because we never use it as alignment character, which is something typical TEX. The same is true for ^ and _. So, effectively we have (for instance):

```
\catcode"26=12
```

In order to still get this & supported as alignment character in math mode, we have to jump through some hoops. Think of this (again, don't do this in ConTEXt):

```
\bgroup
    \global\mathcode"26="8000

    \catcode"26=4

    \xdef\normalmathaligntab{&}

    \catcode"26=13

    \global\everymath{\def&{\normalmathaligntab}}
\egroup
```

Before we go on you should realize that we never use the `&` in ConTeXt as separator. The sole reason for dealing with this issue is that users can have their own code that uses the ampersand that way. In ConTeXt we do things like:

```
\startformula
    \startmatrix
        \NC 1 \NC 2 \NR
        \NC 3 \NC 4 \NR
    \stopmatrix
\stopformula
```

Where `\NC` can be more powerful than a `&`. Anyhow, the reason for discussing this here is that there can be surprises. In a running text you can do this:

```
A & B
```

Which procces okay and gives the ampersand as glyph. The following is also okay:

```
$A \Umathchar"2"0"26 B$
```

However, the next one:

```
$A \char"26 B$
```

fails with a `Misplaced alignment tab character &`. The reason is that where in text mode TeX's parser will turn the `\char` into a character node and carry on afterwards, in math mode it will treat this inpout as were it a directly input character, so the above is like, where the & has active properties and becomes the sparator ampersand which then triggers the error:

```
$A & B$
```

This means that we cannot have a definition like:

```
\def\AND{\char"26\relax}
```

that can be used in math mode, which is why the cweb macros do:

```
\def\AND{\def\AND{\mathchar"2026\relax}\AND}
```

Back to the plain example. The `\matrix` command has to be wrapped in math mode and therefore the & will adapt, while in most ConTeXt constructs that use alignment, we're not in math mode at all when we start with the alignment. Therefore the & will be just an ampersand in most ConTeXt cases.

So to summarize: don't expect `\char"26` to work out well in math mode because all kind of magic kicks in. These are the more obscure features and side effects of TEX dealing with input and it's really hard to predict how TEX will see the ampersand you entered. You need to know the internals and even then it's non trivial. Take

```
\startformula
\startalign
    \NC x \NR
    \NC x \NR
\stopalign
\stopformula
```

versus:

```
\startformula
\startalign
    & x \NR
    & x \NR
\stopalign
\stopformula
```

versus:

```
\startformula
\startalign
    \NC x & y \NR
    \NC x & y \NR
\stopalign
\stopformula
```

The first case works as expected, the second one treats the `&` as text and the third one, as we enter math mode with `\NC`, depends on circumstances. If you use just ConTEXt math coding, you can say:

```
\setupmathematics
  [ampersand=normal]
```

And always render an ampersand (although a math one in math mode).

12 Grouping

12.1 Some details

In TEX there are all kind of groups. When you start with a curly brace, you often enter a group but when you start a box or table cell you also do that. When you enter math a math group is started. Assignments are, unless explicitly done global, nearly always local to the group. Here we discuss the following two cases:

```
{ .... }
\bgroup .... \egroup
\begingroup .... \endgroup
```

We say two cases, not three, because the first two are equivalent: the two macros in the second line are not primitives but aliases to the curly braces. There is however one fundamental difference between them. The verbose `\begingroup` starts a so called simple group so let's call the other complex. A complex group is bounded by equivalents to the two characters (braces) that have catcodes that begin and end these groups. So, the following is valid.

```
{ .... }
{ .... \egroup
\bgroup .... }
\bgroup .... \egroup
```

This means that a macro like this is okay:

```
\def\foo{\bgroup\bf\let\next}  \foo{text}
```

The `\foo` will start a complex group, switch font and then pick up the brace. The group will be closed by the matching right brace or an equivalent. The following two cases are invalid:

```
\bgroup .... \endgroup
\begingroup .... \egroup
```

which means that:

```
\def\foo{\begingroup\bf\let\next}  \foo{text}
```

will trigger an error message. This is rather unfortunate because using braces to wrap an argument in curly braces is rather convenient. The way out is this:

```
\def\foo#1{\begingroup\bf#1\endgroup}  \foo{text}
```

which is perfectly valid apart from the fact that the argument is first picked up and then fed back into the input. Apart from a small performance hit and using more memory it also adds noise to tracing.

12.2 Side effects

In math mode matters are complicates by the fact that complex groups (the ones started with the curly brace) start a math list. And that has side effects because the spacing between math elements depends on what we deal with: math symbol, lists, fenced material. The following example shows a whole lot of this:

```
\starttabulate[|j1|j1||]
    \NC $\sin{\left(xxx\right)}$
    \NC $f   {\left(xxx\right)}$
    \NC $x   {\left(xxx\right)}$
    \NR
    \NC $\sin\begingroup\left(xxx\right)\endgroup$
    \NC $f   \begingroup\left(xxx\right)\endgroup$
    \NC $x   \begingroup\left(xxx\right)\endgroup$
    \NR
    \NC $\sin\left(xxx\right)$\par
    \NC $f   \left(xxx\right)$\par
    \NC $x   \left(xxx\right)$\par
    \NR
    \NC $\sin\color[darkgreen]{(xxx)}$\par
    \NC $f   \color[darkgreen]{(xxx)}$\par
    \NC $x   \color[darkgreen]{(xxx)}$\par
    \NR
    \NC $\sin\startcolor[darkblue](xxx)\stopcolor$\par
    \NC $f   \startcolor[darkblue](xxx)\stopcolor$\par
    \NC $x   \startcolor[darkblue](xxx)\stopcolor$\par
    \NR
    \NC $\sin(xxx)$\par
    \NC $f   (xxx)$\par
    \NC $x   (xxx)$\par
    \NR
    \NC $\sin{\color[darkyellow]{(xxx)}}$\par
    \NC $f   {\color[darkyellow]{(xxx)}}$\par
    \NC $x   {\color[darkyellow]{(xxx)}}$\par
    \NR
    \NC $\sin\begingroup\color[darkred]{(xxx)}\endgroup$\par
    \NC $f   \begingroup\color[darkred]{(xxx)}\endgroup$\par
    \NC $x   \begingroup\color[darkred]{(xxx)}\endgroup$\par
    \NR
\stoptabulate
```

A valid question is why we would want to add curly braces. One reason is that we we want to apply something to a few characters, in this case color the argument to the sinus. Now, when a color command is defined as the `\foo` before, we start a complex group and that influences spacing! This is demonstrated in the left part of figure 12.1 (you can zoom in on the table to see the details; we also report the kind of spacing applied).

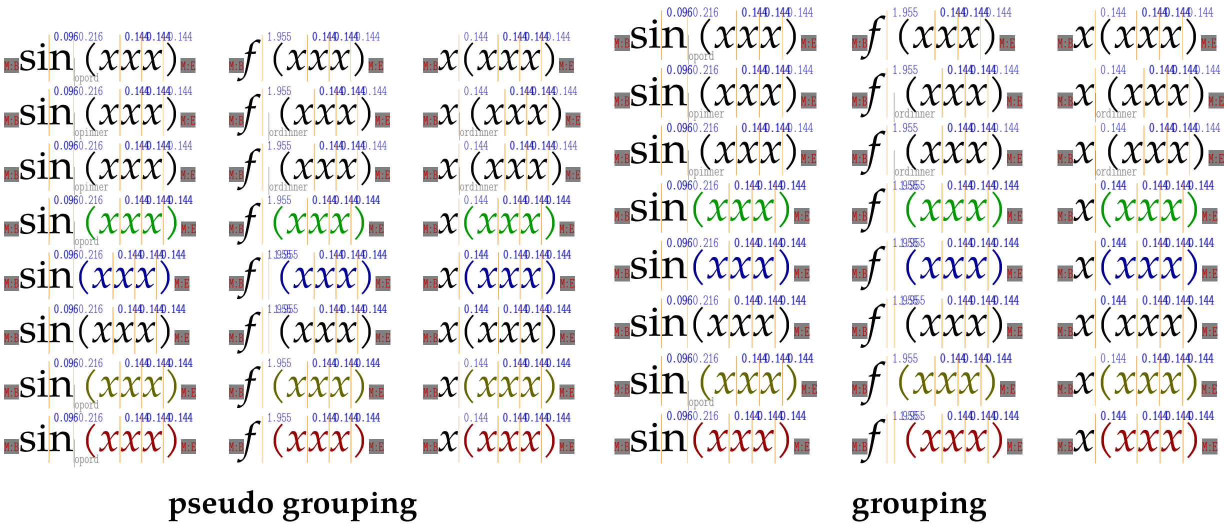


Figure 12.1 Grouping influencing math spacing (list).

The right variant in that figure uses a different way of grouping, one that is equivalent to:

```
\def\foo{\beginsimplegroup\bf\let\next}
```

This time we effectively do `\begingroup` but permits both `\endgroup` or a curly brace (or `\egroup` to wrap up the group. That means that we don't get the side effect of starting a math list.

This example shows the effect of coloring a single character (the result is shown in figure 12.2):

```
$ x = y \quad x \color[red]{=} y$
```

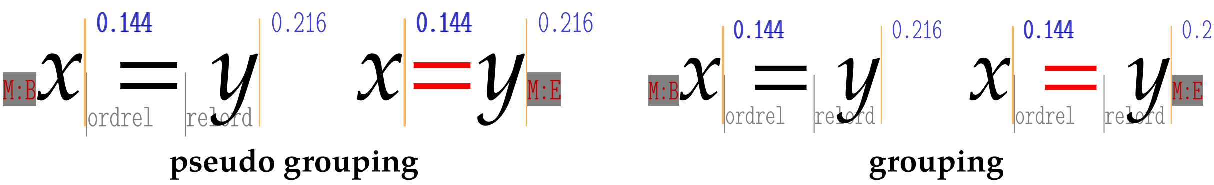


Figure 12.2 Grouping influencing math spacing (symbol).

13 Fun stuff

When I decided to add the \uuid macro (a trivial task because there was already a LUA function for it in CONTEXT) I wondered about other functions that could be added, like those for sine and cosine. There is no real need for that because we can already do this:

```
\luaexpr{math.sin(math.pi/4)}
```

which gives us 0.70710678118655 as result, but still one can ponder the usability of additional macros. When seeing this, one of the first things that probably comes to mind is how to get less digits, and indeed that can be achieved, as 0.7071 demonstrates.

```
\luaexpr[.4N]{math.sin(math.pi/4)}
```

The optional argument between square brackets is a template as we know from other CONTEXT commands without the leading percentage sign. But what if we don't want this expression and explicit math function call?

```
$ \sin (x) = \the\sin{pi/8} $
```

This gives us a normal rendered sin function symbol at the left hand and a numeric result at the right hand: $\sin(x) = 0.382683$. The nice thing about it is that we don't need to come up with new macro names.[2] In a similar fashion we can do this:

```
$ \sind(x) = \luaexpr[.4N]{math.sind(120)} = \the\sind[.4N]{120} $
```

Both calls give the same result: $(x) = 0.866 = 0.866$ and in case you wonder why we have only three digits: the N formatter removes trailing zeros. However, the \the prefix is still not that nice, apart from the fact that we abuse a feature of the LUA interface meant for other purposes (read: we cheat). So, in a next step in exploring this I cooked up:

```
$ \sqrt(x) = \the\sqrt[.3N]     {2} $
$ \sqrt(x) = \compute\sqrt[.3N] {2} $
```

There is still a prefix but \compute looks more natural. It is not an alias for \the but a shortcut for a prefix feature that can drive all kind of interpretations of in this case \sin, and that is probably where the real fun will start. Instead of functions we can also have constants:

In case you wonder how extensible this mechanism is, here is what happens in the mathfun module that needs to be loaded in the usual way. There you find:

The module also provides a few more expression variants (these can end up in the core if really needed much):

```
$ \pi = \mathexpr[.40N]{pi}               $
```

[2] At some point CONTEXT might introduce a namespace mechanism to deal with possible conflicts between environments.

```
$ \pi = \mathexpr[.80N]{sqrt(11)}      $
$ \pi = \decimalexpr[.80N]{sqrt(11)}   $
$ \pi = \decimalexpr{sqrt(11)}         $
$ c = \complexexpr{123 + new(456,789)} $
```

This gives:

$\pi = 3.1415926535897931$
$\pi = 3.3166247903553998$
$\pi = 3.3166247903553998$
$\pi = 3.3166247903553998491149327366706866839270885455894$
$c = 579 + 789i$

The question is: do we need this and if so, what more do we need? Feel free to bing it up on the ConTEXt mailing list. It anyway is a nice dmonstration of what can be done with the mix of languages.