erw-l3*

Erwann Rogard†

Released 2019/10/12

Abstract

LaTeX3 package defining commands built around expl3[1]. For example, `\erw_compose` implements the mathematical concept $f_1 \circ f_2 \cdots \circ f_n$.

Contents

*This file describes version v0.1.5, last revised 2019/10/12.

†firstname dot lastname AusTria gmail dot com

Conventions

The naming conventions are (loosely) those of LaTeX3. For example, ⟨*cs*⟩ stands for *control sequence*, which is described in [1, Part l3basics].

Requirement

Have `erw-l3.sty` is in the path of the LaTeX engine.

Part I
Usage

In the preamble of `\documentclass`, put:

`\usepackage[`⟨*options*⟩`]{erw-l3}`

1 compose

1.1 backend

`\erw_compose:nV` `\erw_compose:nV{`⟨*cs list*⟩`}`⟨*var*⟩
`\erw_compose:nn`
Implements the mathematical concept $f_1 \circ f_2 \cdots \circ f_n$. See Listing 1

`\erw_compose_c:nV` `\erw_compose_c:nV{`⟨*cs names*⟩`}`⟨*var*⟩
`\erw_compose_c:nn`
See Listing 2

`\erw_compose_seq:nV` `\erw_compose_seq:nV{`⟨*cs list*⟩`}`⟨*seq*⟩
Same as `\erw_compose:nV`, but saves each intermediary step See Listing 3

`\erw_compose_seq_c:nV` `\erw_compose_seq_c:nV{`⟨*cs names*⟩`}`⟨*seq*⟩
See Listing 4

`\erw_compose_vers:nV` `\erw_compose_vers:nV{`⟨*list of cs or code*⟩`}`⟨*var*⟩
`\erw_compose_vers:nn`
See Listing 5. Only the `nn` version is implemented

`\erw_compose_seq_vers:nV` `\erw_compose_seq_vers:nV{`⟨*list of cs or code*⟩`}`⟨*seq*⟩
`\erw_compose_seq_vers:nn`
Not implemented

2 csutil

2.1 backend

`\erw_accum:nn` `\erw_accum:nn{`⟨*token list*⟩`}{`⟨*item*⟩`}`
Expands to a token list comprising the items of ⟨*token list*⟩ and ⟨*item*⟩

\erw_apply:Nn
\erw_apply:cn
\erw_apply:Nnn
\erw_apply:Nnnn
\erw_apply:Nnnnn

\erw_apply:Nn⟨*cs*⟩{⟨*arg*⟩}

Expands to ⟨*cs*⟩{⟨*arg*⟩}

\erw_cs_set_eq:NN
\erw_cs_set_eq:cN
\erw_cs_gset_eq:NN
\erw_cs_gset_eq:cN

\erw_cs_set_eq:NN⟨*cs1*⟩⟨*cs2*⟩

⟨*cs1*⟩←⟨*cs2*⟩

\erw_cs_set_inline:Nn
\erw_cs_set_inline:cn
\erw_cs_gset_inline:Nn
\erw_cs_gset_inline:cn

\erw_cs_set_inline:Nn⟨*cs*⟩{⟨*code*⟩}

\erw_identity:n

\erw_identity:n{⟨*arg*⟩}

Expands to ⟨*arg*⟩

\erw_is_matrix_p:n
\erw_is_matrix:nTF

\erw_is_matrix_p:n{⟨*token list*⟩}

Checks if ⟨*token list*⟩ is a (square) matrix.

\erw_fold:NV
\erw_fold:cV

\erw_fold:NV⟨*cs*⟩⟨*var*⟩

⟨*var*⟩←\erw_apply:NV⟨*cs*⟩⟨*var*⟩. See Listing 7.

\erw_last_item:nn

\erw_last_item:nn{⟨*int*⟩}{⟨*token list*⟩}

\erw_merge:nn

\erw_merge:nn{⟨*tl 1*⟩}{⟨*tl 2*⟩}

Merges ⟨*tl 1*⟩⟨*tl 2*⟩

\erw_repeat:nn

\erw_repeat:nn{⟨*int*⟩}{⟨*value*⟩}

See Listing 9

\erw_split:nn

\erw_split:nn{⟨*token list*⟩}{⟨*delimiter*⟩}

See Listing 10

3 int

3.1 backend

\erw_int_range:nn

\erw_int_range:nn{⟨*first*⟩}last

Returns a range of integers. Implementation different than \int_step_inline

\erw_int_range:n

\erw_int_range:n{⟨*count*⟩}

Returns a range of integers. Implementation different than \int_step_inline. See Listing 11

4 map

4.1 backend

\erw_set_map:N
\erw_gset_map:N
\erw_set_map:N⟨cs⟩
Sets the function used by \erw_map:n.

\erw_set_map_inline:n
\erw_gset_map_inline:n
\erw_set_map_inline:n{⟨code⟩}
Sets the function used by \erw_map:n.

\erw_map:n
\erw_map:n{⟨token list⟩}
Applies the stored ⟨cs⟩ to each item in ⟨token list⟩. An application is \erw_is_matrix

\erw_map:Nn
\erw_map:Nn⟨cs⟩{⟨token list⟩}
See Listing 12. Redundant with \tl_map_function:nN

\erw_map_inline:nn
\erw_map_inline:nn{⟨code⟩}{⟨args⟩}
See Listing 13

\erw_map_indexed:Nnn
\erw_map_indexed:Nnn⟨cs⟩{⟨int⟩}{⟨matrix of tokens⟩}
Not implemented. See Listing 15.

\erw_map_thread:Nn
\erw_map_thread:Nn⟨cs⟩{⟨matrix of tokens⟩}
Threads ⟨cs⟩ over the columns, where the arity of ⟨cs⟩ must be equal to the number of rows. See Listing 14

\erw_map_thread_at:Nnn
\erw_map_thread_at:Nnn⟨cs⟩{⟨matrix of tokens⟩}

5 numbrdcs

5.1 backend

\erw_numbrd_cs_reset:
\erw_numbrd_cs_reset:{}
See Listing 16

\erw_numbrd_cs_new:n
\erw_numbrd_cs_new:n {⟨cs or code⟩}
Use it as the first arg to \tl_function_map:Nn

\erw_numbrd_cs:nn
\erw_numbrd_cs:nn {⟨cs or code⟩}

\erw_numbrd_cs_names_braced:nnn
\erw_numbrd_cs_names_braced:nnn{⟨first⟩}{⟨step⟩}{⟨last⟩}
See Listing 16

5.2 frontend

`\numbrdcsnew` `\numbrdcsnew*`

`\numbrdcsnew{`⟨*list of cs or code*⟩`}`

Creates numbered control sequences. The starred version does not reset. See Listing 17

`\numbrdcs`

`\numbrdcs{`⟨*int*⟩`}{`⟨*arg*⟩`}`

Evaluates control sequence numbered ⟨*int*⟩ with argument ⟨*arg*⟩. See Listing 17

Part II
Listings

1 compose

1.1 backend

Listing 1

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\cs_set:Npn \__bar #1 {g[#1]}
\cs_set:Npn \__baz #1 {h\{#1\}}
\tl_set:Nn \l_tmpa_tl{X}
\erw_compose:nV{
  {\__baz}{\__bar}{\__foo}}
  \l_tmpa_tl
\l_tmpa_tl                              h{g[f(X)]}
\tl_set:Nn \l_tmpa_tl{X}
\erw_compose:nn{
  {\__baz}{\__bar}{\__foo}}
  {X}                                   h{g[f(X)]}
\ExplSyntaxOff
```

Listing 2

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\cs_set:Npn \__bar #1 {g[#1]}
\cs_set:Npn \__baz #1 {h\{#1\}}
\tl_set:Nn \l_tmpa_tl{X}
\erw_compose_c:nV{
  {__baz}{__bar}{__foo}}
  \l_tmpa_tl
\l_tmpa_tl                              h{g[f(X)]}
\erw_compose_c:nn{
  {__baz}{__bar}{__foo}}
  {X}                                   h{g[f(X)]}
\ExplSyntaxOff
```

Listing 3

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\cs_set:Npn \__bar #1 {g[#1]}
\cs_set:Npn \__baz #1 {h\{#1\}}
\seq_new:N\l_tmp_seq
\seq_put_right:Nn\l_tmp_seq{X}
  \erw_compose_seq:nV{
    {\__baz}{\__bar}{\__foo}}
    \l_tmp_seq
\seq_item:Nn\l_tmp_seq{1}           X
\seq_item:Nn\l_tmp_seq{2}           f(X)
\seq_item:Nn\l_tmp_seq{3}           g[f(X)]
\seq_item:Nn\l_tmp_seq{4}           h{g[f(X)]}
\ExplSyntaxOff
```

Listing 4

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\cs_set:Npn \__bar #1 {g[#1]}
\cs_set:Npn \__baz #1 {h\{#1\}}
\seq_new:N\l_tmp_seq
\seq_put_right:Nn\l_tmp_seq{X}
\erw_compose_seq_c:nV{
  {__baz}{__bar}{__foo}}
  \l_tmp_seq
\seq_item:Nn\l_tmp_seq{1}           X
\seq_item:Nn\l_tmp_seq{2}           f(X)
\seq_item:Nn\l_tmp_seq{3}           g[f(X)]
\seq_item:Nn\l_tmp_seq{4}           h{g[f(X)]}
\ExplSyntaxOff
```

Listing 5

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\cs_set:Npn \__bar #1 {g[#1]}
\cs_set:Npn \__baz #1 {h\{#1\}}
\erw_compose_vers:nn{
  {\__baz}{g[#1]}{\__foo}}
  {X}                               h{g[f(X)]}
\ExplSyntaxOff
```

2 csutil

2.1 backend

Listing 6

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\erw_apply:Nn\__foo{X}                f(X)
\ExplSyntaxOff
```

Listing 7

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {f(#1)}
\tl_set:Nn \l_tmpa_tl{X}
\erw_fold_set_par:n{Nf}
\erw_fold_apply_par:n{Nf}
\erw_fold:NV\__foo\l_tmpa_tl
\l_tmpa_tl                            f(X)
\cs_set:Npn\__bar #1 {g[#1]}
\erw_fold:cV{__bar}\l_tmpa_tl
\l_tmpa_tl                            g[f(X)]
\ExplSyntaxOff
```

Listing 8

```
\ExplSyntaxOn
\erw_is_matrix:nTF
{
        { {a}{b}{c} }
        { {k}{l}{m} }
        { {x}{y}{z} }
}{T}{F}                               T
\erw_is_matrix:nTF
{
        { {a}{c} }
        { {k} }
        { {x}{y}{z} }
}{T}{F}                               F
\ExplSyntaxOff
```

Listing 9

```
\ExplSyntaxOn
\erw_repeat:nn
  {3}{abracad}abra                    abracadabracadabracadabra
\ExplSyntaxOff
```

Listing 10

```
\ExplSyntaxOn
\erw_split:nn
  {{a}{b}{c}}{==}                    a==b==c
\ExplSyntaxOff
```

3 int

3.1 backend

Listing 11

```
\ExplSyntaxOn
\erw_int_range:nn{2}{5}              2345
\erw_int_range:n{5}                  12345
\ExplSyntaxOff
```

4 map

4.1 backend

Listing 12

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {(#1)}
\erw_map:Nn \__foo{{a}{b}{c}}        (a)(b)(c)
\ExplSyntaxOff
```

Listing 13

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 {(#1)}
\erw_map_inline:nn{
  (#1)}{{a}{b}{c}}                   (a)(b)(c)
\ExplSyntaxOff
```

Listing 14

```
\ExplSyntaxOn
\cs_set:Npn \__foo:n #1 {(#1)}
\erw_map_thread:Nn \__foo:n
{
     {{a}{b}{c}{d}{e}{f}}
}                                          (a)(b)(c)(d)(e)(f)
\cs_set:Npn \__foo:nn  #1 #2
     {(#1+#2)}
\erw_map_thread:Nn \__foo:nn
{
     {{a}{b}{c}{d}{e}{f}}
     {{A}{B}{C}{D}{E}{F}}
}                                          (a+A)(b+B)(c+C)(d+D)(e+E)(f+F)
\cs_set:Npn \__foo:nnn
     #1 #2 #3
    {(#1+#2+#3)}
\erw_map_thread:Nn \__foo:nnn
{
     {{a}{b}{c}{d}{e}{f}}
     {{A}{B}{C}{D}{E}{F}}
     {{k}{l}{m}{n}{o}{p}}
}                        (a+A+k)(b+B+l)(c+C+m)(d+D+n)(e+E+o)(f+F+p)
\cs_set:Npn \__foo:nnnn
     #1 #2 #3 #4
    {(#1+#2+#3+#4)}
\erw_map_thread:Nn \__foo:nnnn
{
     {{a}{b}{c}{d}{e}{f}}
     {{A}{B}{C}{D}{E}{F}}
     {{k}{l}{m}{n}{o}{p}}
     {{K}{L}{M}{N}{O}{P}}
}       (a+A+k+K)(b+B+l+L)(c+C+m+M)(d+D+n+N)(e+E+o+O)(f+F+p+P)
\ExplSyntaxOff
```

Listing 15 Debugging for `\erw_map_indexed`

```
\ExplSyntaxOn
\cs_set_protected:Npn \__foo:nn #1 #2
    {(#1+#2)}
\erw_map_thread:Nn
    \__foo:nn
        {
            {{1}{2}{3}}
            {{a}{b}{c}}
        }                                   (1+a)(2+b)(3+c)
\exp_last_unbraced:Nx
\erw_map_thread:Nn
{
        \__foo:nn
        {
            {\erw_int_range:n{3}}
            {{a}{b}{c}}
        }
}                                           (123+a)          (does not thread!)
\exp_last_unbraced:Nx
\erw_map_thread:Nn
{
        \__foo:nn
        {
            {\int_step_inline:nn{3}{#1}}
            {{a}{b}{c}}
        }
}               Illegal parameter number in definition of \l__exp_internal_tl!
\ExplSyntaxOff
```

5 numbrdcs

5.1 backend

Listing 16

```
\NewDocumentCommand{\myfoo}{m}{f(#1)}
\NewDocumentCommand{\mybar}{m}{g[#1]}
\NewDocumentCommand{\mybaz}{m}{h\{#1\}}
\numbrdcsnew{{\mybaz}{g[#1]}{\myfoo}}
\ExplSyntaxOn
\exp_last_unbraced:Nx
  \erw_compose_c:nn
  {
    {\erw_numbrd_cs_names_braced:
         nnn{1}{1}{3}}
    {X}
  }
\ExplSyntaxOff                              h{g[f(X)]}
```

5.2 frontend

Listing 17

```
\NewDocumentCommand{\thefoo}{m}{f(#1)}
\NewDocumentCommand{\thebar}{m}{g[#1]}
\NewDocumentCommand{\thebaz}{m}{h\{#1\}}
\numbrdcsnew{
    {\thefoo}
    {g[#1]}
    {\thebaz}}
\numbrdcs{1}{X}                    f(X)
\numbrdcs{2}{X}                    g[X]
\numbrdcs{3}{X}                    h{X}
\numbrdcsnew*{
    {\thefoo}
    {g[#1]}
    {\thebaz}}
\numbrdcs{4}{X}                    f(X)
\numbrdcs{5}{X}                    g[X]
\numbrdcs{6}{X}                    h{X}
```

Part III
Implementation

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}[2018/06/01]
\RequirePackage{xparse}[2018/02/01]
\RequirePackage{l3keys2e}
\ExplSyntaxOn
\msg_new:nnn{erw}{generic}{#1}
```

1 compose

1.1 backend

```
\cs_set:Npn \erw_compose:NnV
  #1 % method
  #2 % funs
  #3 % var
{
  \erw_fold_set_par:n{Nf}
  \erw_fold_apply_par:n{Nf}
  \erw_cs_set_inline:Nn \__erw_map:n
  {
     #1{##1}#3
  }
  \exp_args:Nf\erw_map:n
  {
```

```
      \tl_reverse:n{#2}
  }
}
\cs_set:Npn \erw_compose:nV #1 #2
{
  \erw_compose:NnV \erw_fold:NV {#1} #2
}
\cs_set:Npn \erw_compose_c:nV #1 #2
{
  \erw_compose:NnV \erw_fold:cV {#1} #2
}
\tl_new:N \__erw_compose_tl
\cs_set:Npn \erw_compose:nn #1 #2
{
  \tl_set:Nn \__erw_compose_tl {#2}
  \erw_compose:nV{#1}\__erw_compose_tl
  \__erw_compose_tl
}
\cs_set:Npn \erw_compose_c:nn #1 #2
{
  \tl_set:Nn \__erw_compose_tl {#2}
  \erw_compose_c:nV{#1}\__erw_compose_tl
  \__erw_compose_tl
}
\cs_set:Npn \erw_compose_seq:nV #1 #2
{
  \erw_compose:NnV \erw_fold_seq:NV {#1} #2
}
\cs_set:Npn \erw_compose_seq_c:nV
  #1 % funs
  #2 % seq
{
  \erw_compose:NnV \erw_fold_seq:cV {#1} #2
}
\cs_set:Npn \erw_compose_vers:nV #1 #2
{
   \msg_error:nnn{erw}{generic}{erw_compose_vers:nV~yet-to~be~implemented}
}
\cs_set:Npn \erw_compose_seq_vers:nV #1 #2
{
   \msg_error:nnn{erw}{generic}{erw_compose_vers:nV~yet-to~be~implemented}
}
\cs_set:Npn \erw_compose_vers:nn #1 #2
{
   \erw_numbrd_cs_reset:{}
      \tl_map_function:nN{#1}\erw_numbrd_cs_new:n
      \exp_last_unbraced:Nx
      \erw_compose_c:nn
         {{\erw_numbrd_cs_names_braced:{}}}
         {#2}
}
```

2 csutil

2.1 backend

```
\cs_set:Npn \erw_accum:nn #1 #2
{
    {#1{#2}}
}
\cs_set:Npn \__erw_cs_name:N #1
{
    \exp_last_unbraced:Nf \use_i:nnn {\cs_split_function:N #1}
}
\cs_set:Npn \erw_apply:Nn
  #1 % fun
  #2 % tl
{
  #1{#2}
}
\cs_generate_variant:Nn \erw_apply:Nn {No, Nf, Nx, c}
\cs_set:Npn \erw_cs_set_eq:NN #1 #2
{
  \cs_set:Npn #1 ##1{#2{##1}}
}
\cs_generate_variant:Nn \erw_cs_set_eq:NN {cN}
\cs_set:Npn \erw_cs_gset_eq:NN #1 #2
{
  \cs_gset:Npn #1 ##1{#2{##1}}
}
\cs_generate_variant:Nn \erw_cs_gset_eq:NN {cN}
\cs_set:Npn \erw_cs_set_inline:Nn #1 #2
{
  \cs_set:Npn #1 ##1{#2}
}
\cs_generate_variant:Nn \erw_cs_set_inline:Nn {cn}
\cs_set:Npn \erw_cs_gset_inline:Nn #1 #2
{
  \cs_gset:Npn #1 ##1{#2}
}
\cs_generate_variant:Nn \erw_cs_gset_inline:Nn {cn}
\tl_set:Nn \__erw_fold_set_par_tl{\c_novalue_tl}
\tl_set:Nn \__erw_fold_apply_par_tl{\c_novalue_tl}
\cs_set:Npn \erw_fold_set_par:n #1
{
  \tl_set:Nn \__erw_fold_set_par_tl{#1}
}
\cs_set:Npn \erw_fold_apply_par:n #1
{
  \tl_set:Nn \__erw_fold_apply_par_tl{#1}
}
\cs_set:Npn \erw_fold:NV
  #1 % fun
  #2 % var
{
  \use:c{tl_set:\__erw_fold_set_par_tl}
```

```
     #2
    {\use:c{erw_apply:\__erw_fold_apply_par_tl}{#1}{#2}}
}
\cs_generate_variant:Nn \erw_fold:NV {cV}
\tl_new:N \__erw_fold_seq_item_tl
\cs_set:Npn \erw_fold_seq:NV
  #1 % fun
  #2 % seq
{
  \seq_get_right:NN #2 \__erw_fold_seq_item_tl
  \erw_fold:NV #1 \__erw_fold_seq_item_tl
  \seq_put_right:No #2 {\__erw_fold_seq_item_tl}
}
\cs_generate_variant:Nn \erw_fold_seq:NV {cV}
\cs_set:Npn \erw_identity:n #1{#1}
\prg_set_conditional:Npnn \erw_is_matrix:n #1 { p, TF }
{
    \erw_gset_map_inline:n{==\tl_count:n{##1}}
    \int_compare:nTF
    {
        \exp_args:Nf\tl_count:n{\tl_head:n{#1}}
        \exp_args:Nf \erw_map:n
        {
                \tl_tail:n{#1}
            }
    }
    {\prg_return_true:}
    {\prg_return_false:}
}
% Deprecated in v0.1.4 after realizing \cs{tl_range:n} does the job
%\cs_set:Npn\__erw_items_to:nnn #1 #2 #3
%{
%    \int_compare:nNnTF
%    {#1}>{#2}
%    {
%        \exp_args:Nf \tl_head:n{#3}
%        \__erw_items_to:nnn
%            {#1}
%            {\int_eval:n{#2+1}}
%            {\exp_args:Nf \tl_tail:n{#3}}
%    }
%    {
%        \exp_args:Nf \tl_head:n{#3}
%    }
%}
%\cs_set:Npn \erw_items_to:nn #1 #2
%{
%    \__erw_items_to:nnn
%        {#1}
%        {1}
%        {#2}
%}
\cs_set:Npn \erw_last_item:n #1
{
```

```
    \exp_args:Nof \tl_item:nn
        {#1}
        {
            \tl_count:n{#1}
        }
}
\cs_set:Npn \erw_merge:nn #1 #2
{
    {#1#2}
}
\cs_set:Npn \erw_repeat:nn #1 #2
{
    \int_step_inline:nnnn{1}{1}{#1}{#2}
}
\cs_set:Npn \erw_split:nnn #1 #2 #3
{
    \tl_head:n{#1}
    \use:c{exp_args:#3} \tl_map_inline:nn
    {
        \tl_tail:n
        {
            #1
        }
    }{#2##1}
}
\cs_set:Npn \erw_split:nn #1 #2
{
    \erw_split:nnn{#1}{#2}{Nf}
}
```

3 map

3.1 backend

```
\cs_set:Npn \__erw_int_range:nnn #1 #2 #3
{
    \int_compare:nNnTF
    {
            \int_eval:n{#2+1}
    }>{#3}
    {
        {#1}
    }
    {
        \__erw_int_range:nnn
        {
            \exp_args:Nx\erw_accum:nn{#1}
            {
                \int_eval:n{#2+1}
            }
        }
        {\int_eval:n{#2+1}}
        {#3}
    }
```

```
}
\cs_set:Npn \erw_int_range:nn #1 #2
{
    \__erw_int_range:nnn {{#1}}{#1}{#2}
}
\cs_set:Npn \erw_int_range:n #1
{
    \__erw_int_range:nnn {}{0}{#1}
% Alt to:
%    \int_step_inline:nn {#1}{##1}
}
```

4 map

4.1 backend

```
\cs_set:Npn \erw_gset_map:N #1
{
    \erw_cs_gset_eq:NN \__erw_map:n #1
}
\cs_set:Npn \erw_gset_map_inline:n #1
{
    \erw_cs_gset_inline:Nn \__erw_map:n {#1}
}
\cs_set:Npn \erw_map:n #1
{
  \__erw_map:nn#1\q_recursion_tail\q_recursion_stop\q_recursion_tail\q_recursion_stop
}
\cs_set:Npn \__erw_map:nn #1 #2
{
  \quark_if_recursion_tail_stop:n{#1}
  \__erw_map:n{#1}  \__erw_map:nn{#2}
}
\cs_new:Npn \__erw_map:n #1
{
  \msg_error:nnn
    {erw}
    {generic}
    {__erw_map:n~not~set}
}
\cs_set:Npn \erw_map:Nn
  #1 % fun
  #2 % tl
{
  \erw_cs_set_eq:NN \__erw_map:n #1
  \erw_map:n{#2}
}
\cs_set:Npn \erw_map_inline:nn
  #1 % inl
  #2 % tl
{
  \erw_cs_set_inline:Nn \__erw_map:n {#1}
  \erw_map:n{#2}
}
```

```
\cs_set:Npn \erw_apply:Nnn #1 #2 #3
{
    #1{#2}{#3}
}
\cs_set:Npn \erw_apply:Nnnn #1 #2 #3 #4
{
    #1{#2}{#3}{#4}
}
\cs_set:Npn \erw_apply:Nnnnn #1 #2 #3 #4 #5
{
    #1{#2}{#3}{#4}{#5}
}
\cs_set:Npn \__erw_map_thread_at:Nnn #1 #2 #3
{
     \erw_apply:Nn #1
     {\exp_args:Nf\tl_item:nn {#3} {#2} }
}
\cs_set:Npn \__erw_map_thread_at:Nnnn #1 #2 #3 #4
{
     \erw_apply:Nnn #1
     {\exp_args:Nf\tl_item:nn {#3} {#2} }
     {\exp_args:Nf\tl_item:nn {#4} {#2} }
}
\cs_set:Npn \__erw_map_thread_at:Nnnnn #1 #2 #3 #4 #5
{
     \erw_apply:Nnnn #1
     {\exp_args:Nf\tl_item:nn {#3} {#2} }
     {\exp_args:Nf\tl_item:nn {#4} {#2} }
     {\exp_args:Nf\tl_item:nn {#5} {#2} }
}
\cs_set:Npn \__erw_map_thread_at:Nnnnnn #1 #2 #3 #4 #5 #6
{
     \erw_apply:Nnnnn #1
     {\exp_args:Nf\tl_item:nn {#3} {#2} }
     {\exp_args:Nf\tl_item:nn {#4} {#2} }
     {\exp_args:Nf\tl_item:nn {#5} {#2} }
     {\exp_args:Nf\tl_item:nn {#6} {#2} }
}
\cs_set:Npn \erw_map_thread_at:Nnn #1 #2 #3
{
    \exp_args:Nf\int_case:nnTF
    {
        \tl_count:n{#3}
    }
    {
        {1}{ \__erw_map_thread_at:Nnn #1{#2}#3 }
        {2}{ \__erw_map_thread_at:Nnnn #1{#2}#3 }
        {3}{ \__erw_map_thread_at:Nnnnn #1{#2}#3 }
        {4}{ \__erw_map_thread_at:Nnnnnn #1{#2}#3 }
    }
    {
        % Do nothing
    }
    {
```

```
        \msg_error:nnn{erw}
            {generic}
            {erw_map_thread_at:~count~of~#3~not~withing~1~to~4}
    }
}
\cs_set:Npn \erw_map_thread:Nn #1 #2
{
    % TODO check that #2 is a matrix
    \int_step_inline:nn
    {
        \exp_args:Nf \tl_count:n{ \tl_head:n{#2} }
    }
    {
        \erw_map_thread_at:Nnn #1 {##1} {#2}
    }
}
```

5 numbrdcs

5.1 backend

```
\int_new:N \__erw_numbrd_cs_int
\cs_set:Npn \erw_numbrd_cs_name:n #1{__erw_numbrd_cs_\int_to_alph:n{#1}:n}
\cs_set:Npn \erw_numbrd_cs_name_braced:n #1{{\erw_numbrd_cs_name:n{#1}}}
\tl_set:Nn \__erw_numbrd_cs_name_tl {\erw_numbrd_cs_name:n{\__erw_numbrd_cs_int}}
\cs_set:Npn \erw_numbrd_cs:nn #1 #2
{
    \erw_apply:cn{__erw_numbrd_cs_\int_to_alph:n{#1}:n}{#2}
}
\cs_new_protected:Npn \erw_numbrd_cs_reset:
{
    \int_zero:N \__erw_numbrd_cs_int
    \tl_set:Nn \__erw_numbrd_cs_ext_tl{}
}
\cs_new_protected:Npn \erw_numbrd_cs_new:n #1
{
    \int_incr:N \__erw_numbrd_cs_int
    \erw_cs_set_inline:cn{\__erw_numbrd_cs_name_tl}
    {
        \token_if_cs:NTF
            {#1}
            {#1{##1}}
            {#1}
    }
}
\cs_new:Npn \erw_numbrd_cs_names:nnn #1 #2 #3
{
    \int_step_function:nnnN { #1 }{ #2 }{ #3 } \erw_numbrd_cs_name:n
}
\cs_new:Npn \erw_numbrd_cs_names_braced:nnn #1 #2 #3
{
    \int_step_function:nnnN { #1 }{ #2 }{ #3 } \erw_numbrd_cs_name_braced:n
    % TODO \tl_range_braced:nnn?
}
```

```
\cs_new:Npn \erw_numbrd_cs_names_braced:
{
    \erw_numbrd_cs_names_braced:nnn{1}{1}{\__erw_numbrd_cs_int}
}
```

5.2 frontend

```
\NewDocumentCommand{\numbrdcsnew}{ s m }
{
    \IfBooleanTF{#1}
        {}
        { \erw_numbrd_cs_reset:{}}
    \tl_map_function:nN {#2}\erw_numbrd_cs_new:n
}
\NewDocumentCommand{\numbrdcs}{ m m }
{
    \erw_numbrd_cs:nn{#1}{#2}
}
% \ProcessKeysPackageOptions{ erw }
\ExplSyntaxOff
```

Part IV
Other

1 Support

This package is available from https://www.ctan.org/pkg/erw-l3 (release) or https://github.com/rogard/erw-l3 (development) where you can report issues.

2 To do

- Missing variants of `\erw_compose`
- `\erw_map_indexed`. See Listing 15
- Need to give some thought to 'protected'

3 Acknowledgment

I thank those that have answered my questions on forums pertaining to LaTeX3. See here: https://tex.stackexchange.com/users/112708/erwann?tab=questions and here: https://latex.org/forum/memberlist.php?mode=viewprofile&u=61329

References

[1] The LaTeX3 Project Team *The LaTeX3 interfaces* http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf

[2] The LaTeX3 Project Team *The xparse package* http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf

Change History

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

K

M

N

P

Q

R

S

T

U